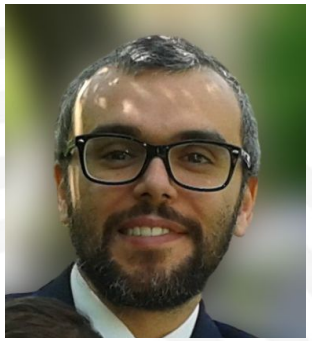
Larger-scale model training on multi-GPU systems

ELLIS Summer School
on
Large-Scale AI for Research and Industry

Modena
18-22 September 2023

Sergio Orlandini
CINECA
s.orlandini@cineca.it

About Me



I am a Senior High Performance Computing specialist at CINECA.

Wide experience in scientific software development on hybrid architectures CPU-GPU, deep learning workloads on HPC systems, GPU programming, optimization and parallelization of scientific and technical applications and evaluation of emerging architectures prototypes.

Many years of experience and collaboration with both Academic Community and Industry Partners.

I am in the advisory board of NVIDIA Artificial Intelligence Technology Centre Italy (NVAITC).

I am also involved in many courses at CINECA on GPU programming, AI & ML, optimization and parallelization of HPC applications.

Tutorial Materials

All materials and slides are available at `hpc-dl-ellis-2023` git repository in the HPC gitlab of CINECA:
https://gitlab.hpc.cineca.it/gfiamen1/hpc-dl-ellis-2023

```
git clone https://gitlab.hpc.cineca.it/gfiamen1/hpc-dl-ellis-2023.git
```

NB: Some datasets are needed for the hands-on sessions. You can copy the required datasets from a shared directory to your local directory with the following command:

```
cp -r /leonardo/pub/userinternal/sorland2/hpc-dl-ellis-2023 hpc-dl-ellis-2023
```

Gentle Course Starting Poll

http://etc.ch/z5id

CINECA

CINECA is the **Italian Supercomputing Center** (https://www.cineca.it).

CINECA is a not-for-profit Consortium, made up of 98 members: the **Italian Ministry of Education**, the **Italian Ministry of Universities and Research**, 69 **Italian universities** and 27 **Italian National Institutions**.

CINECA is currently one of the Large Scale Facilities in Europe and it is a PRACE **Tier-0** hosting site.

The mission of CINECA HPC Department is to accelerate the scientific discovery by providing high performance computing resources, data management and storage systems and tools and HPC services and expertise at large, aiming to develop and promote technical and scientific services related to high-performance computing for the Italian and European research community.

CINECA has five locations in Italy.
The **High Performance Computing** (HPC) Department is dislocated around 3 cities (Bologna, Rome and Milan)

CINECA offers HPC courses and Summer Schools and it is a EURO-HPC Training Centres.
see **CINECA Academy** website for a list of all courses @ https://eventi.cineca.it/en

It is possible to ask for system resource allocations for projects in any scientific discipline applying to targete programs (**Iscra** and **EURO-HPC**) and that are assigned after peer review evaluation, or through specific conventions or industrial contracts.

To use these facilities is necessary/mandatory to become Cineca user

HOW-TO get HPC Resources @CINECA

ISCRA (Italian SuperComputing Resource Allocation)
https://www.hpc.cineca.it/services/iscra

- **ISCRA C**
  - **Small** size projects (10K GPU hours **@LEONARDO**, 100K Core hours **@G100**)
  - One call every month (continuous submission: excluding August and December)
  - Duration 9 months
  - Easy to submit (few data needed)
  - Only **technical** evaluation (internal @CINECA)
  - Ready to go in 1/2 months
- **ISCRA B**
  - **Mid/Large** size projects (up to 250K GPU hours **@LEONARDO**, up to 3.6M Core hours **@G100**, CLOUD, QUANTUM)
  - One call every 6 months
  - Duration 1 year
  - More detailed proposal (tech and scientific details, scalability data, detailed budget estimation)
  - **Technical** and **Scientific** evaluations.
  - Ready to go 4/5 months
- **TRY project**
  - Only for test purposes
  - Few hours for 2 months maximum
  - To be requested at iscra@cineca.it

CINECA GPU-Hackathon

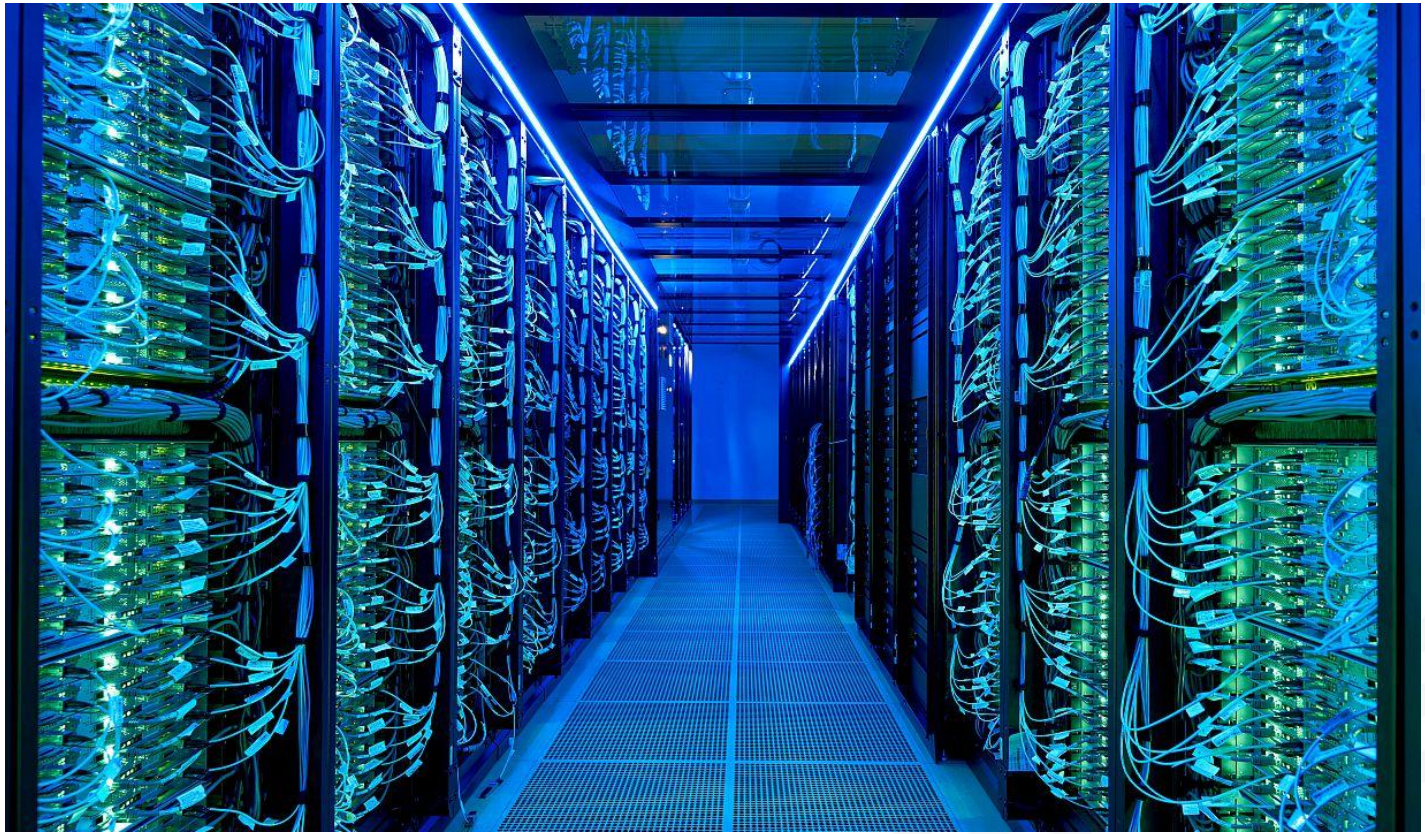
We are happy to announce that the registration is now open for

CINECA GPU-HACKATHON-2023

https://www.openhackathons.org/s/siteevent/a0C5e000005V8ZwEAK/se000156

Stay tune @https://www.openhackathons.org/s

Supercomputers



What is a Supercomputer?

A supercomputer is a class of extremely **powerful computers**. Powerful in terms of computational throughput, memory capacity and accuracy on numerical problem.

Unlike traditional computers, supercomputers are made up of more than one central processing unit (CPU). A supercomputer is a collection of small computers (**compute nodes**) hooked together by an high-speed **interconnection network** plus an I/O system.

A modern supercomputer can contain more than **ten thousands** of nodes. These nodes can collaborate on solving a complex problem reducing overall time to solution.

Supercomputers need **software** that allows the nodes to communicate over the interconnect.

The performance of a supercomputer is measured in floating-point operations per second (**FLOPS**).

Top class supercomputers reach hundreds **PFlops** perfs, one million times more processing power than the fastest laptop.

TOP500 Project

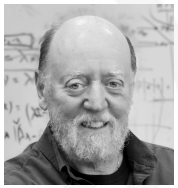
https://www.top500.org

TOP500 is a project to rank the **500 most powerful** non-distributed computer systems in the world.

The project started in 1993 and it is updated every **6 months**:

- International Supercomputing Conference (ISC) in **June**
- ACM/IEEE Supercomputing Conference in **November**

The computer performances are determined by running the **High-Performance LINPACK** (HPL).



- The Linpack Benchmark is a measure of a computer's floating-point rate of execution.
- It solves a dense system of linear equations.
- https://www.top500.org/project/linpack
- "*Performance of Various Computers Using Standard Linear Equations Software*", Jack Dongarra, University of Tennessee, Knoxville, Computer Science Technical Report Number CS - 89 – 85 http://www.netlib.org/benchmark/performance.ps

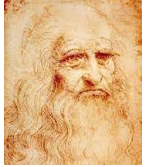
The TOP500 project lists also **Green500** (energy efficiency / performance per watt) and **HPCG benchmark** (stress on the internal interconnect) lists.

TOP500 June 2023

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404

LEONARDO

CINECA

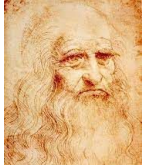


LEONARDO: Pre-Exascale System @CINECA

Italy hosts one of the **pre-exascale** class computers funded by the European Commission in the context of EuroHPC, a joint collaboration between European countries and the European Union about developing and supporting exascale supercomputing.

LEONARDO project was presented by Cineca representing Italy in the agreement conceived and managed by **Cineca**, with the **Italian Ministry of Education, University and Research**, the National Institute of Nuclear Physics (**INFN**) and the International School of Advanced Studies (**SISSA**) and approved by the European Joint Undertaking EuroHPC.

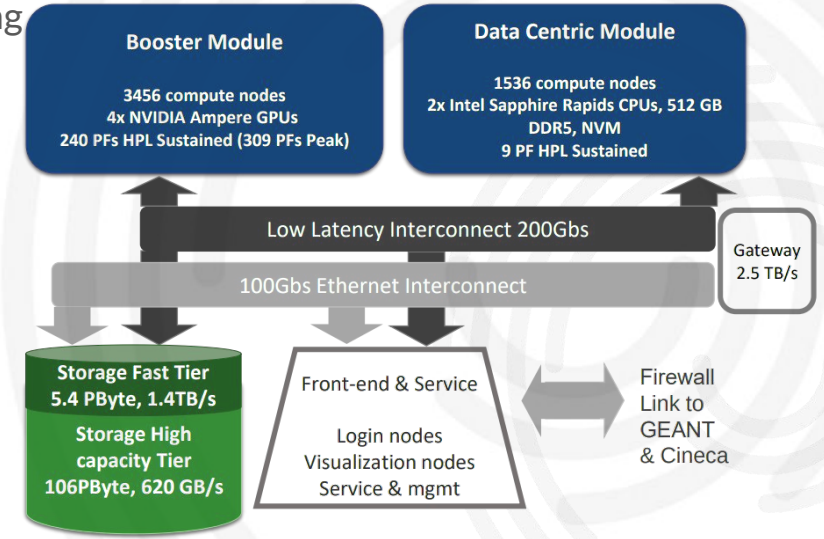
LEONARDO, the new pre-exascale supercomputer is available for production from **August 2023**.

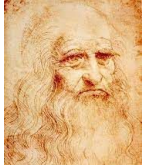


LEONARDO: Pre-Exascale System @CINECA

CINECA's Leonardo will be the fastest European technology based supercomputer: based on **Atos BullSequana XH2000** technology and feature nearly 14,000 next generation **NVIDIA Ampere** architecture-based GPUs and **Mellanox HDR InfiniBand**. As part of EuroHPC, Leonardo will deliver world-class supercomputing capabilities to support HPC and AI Research in Italy and Europe.

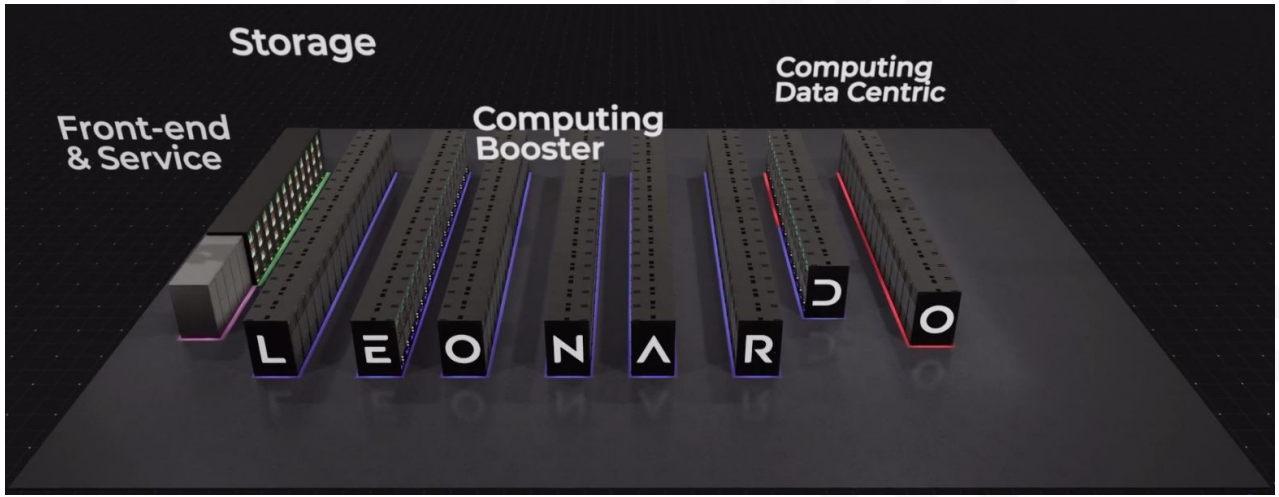
- More than 120 BullSequana XH2000 Direct Liquid cooling racks
- **~5000** computing nodes
  - 3456 GPU nodes (Intel Ice Lake + Nvidia GPUs)
  - 1536 GPP nodes (Intel Sapphire Rapids)
- **14000 GPU**. 4x NVIDIA Ampere GPUs per node
- **300+ PFlops**
- 3+PB RAM
- **120 PB I/O**
- 1 TB/s bandwidth
- 200 Gb/s interconnection **Mellanox Infiniband** HDR
- Dragonfly+ Topology
- **120 Mln €** investment

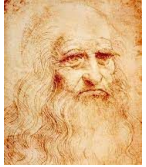




LEONARDO: Pre-Exascale System @CINECA

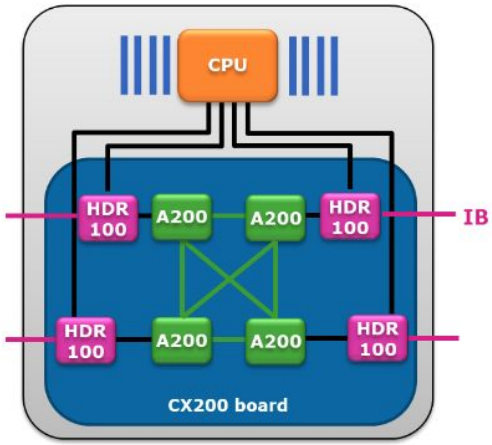
https://www.youtube.com/watch?v=AC4epHSdn0Q&ab_channel=LeonardoPre-exascaleSupercomputer

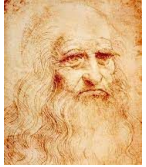




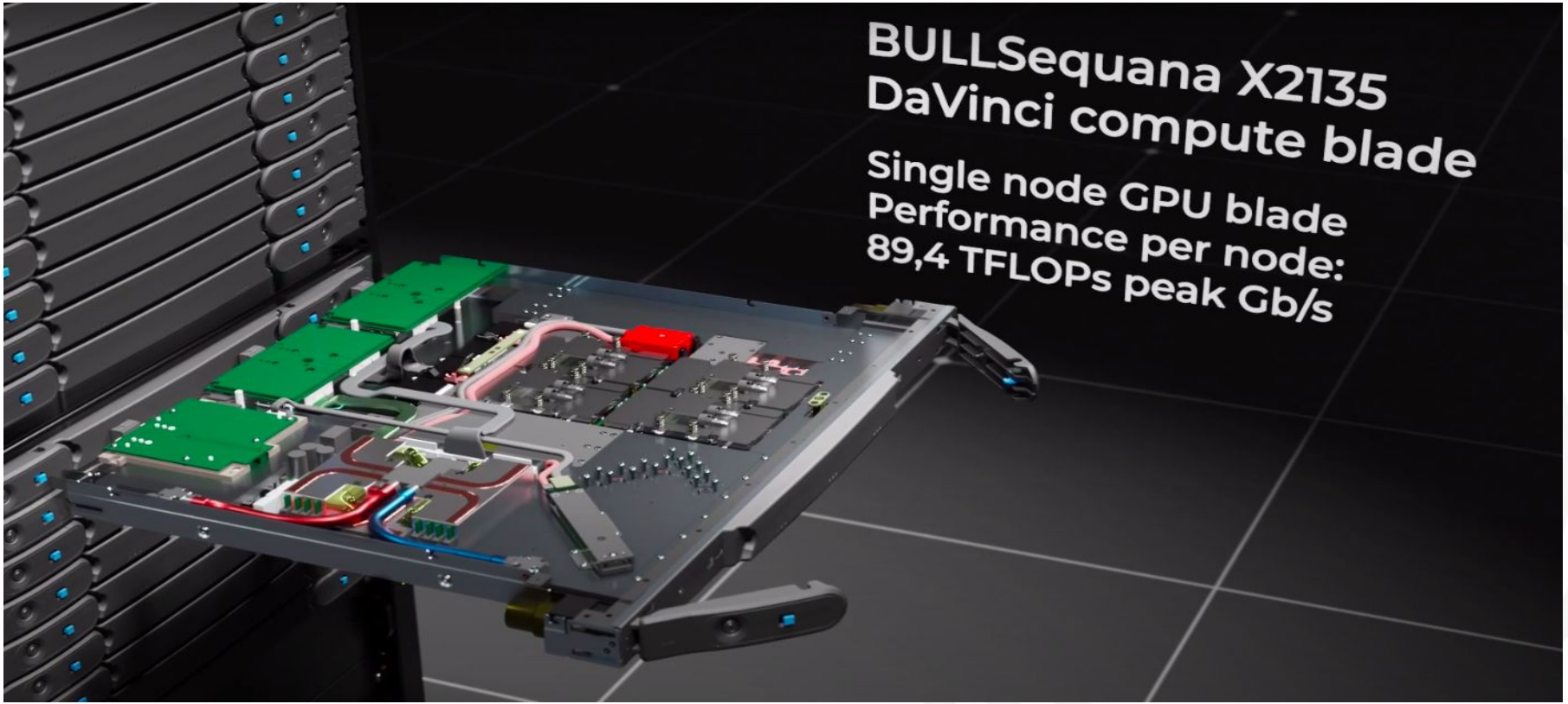
LEONARDO: Booster Module

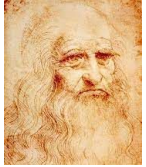
- **3456** GPU nodes + 4 login nodes
- Performances:
  - Peak FP Performance: **309.6 Pflop/s**
  - Linpack (HPL) Performance: 240.5 Pflop/s
  - Performance per node: 89,4 TFLOPs peak
- 1x CPU Intel Xeon 8358 "**IceLake**" (32 cores, 2.6 GHz)
- 512 GB per node: 8x 64 GB RAM DDR4 3200 MHz
- 4x Nvidia Custom Ampere **A100** (64GB HBM2)
- 2x **Mellanox InfiniBand** HDR100/200 2x100 Gb/s
- **Dragonfly+** Topology



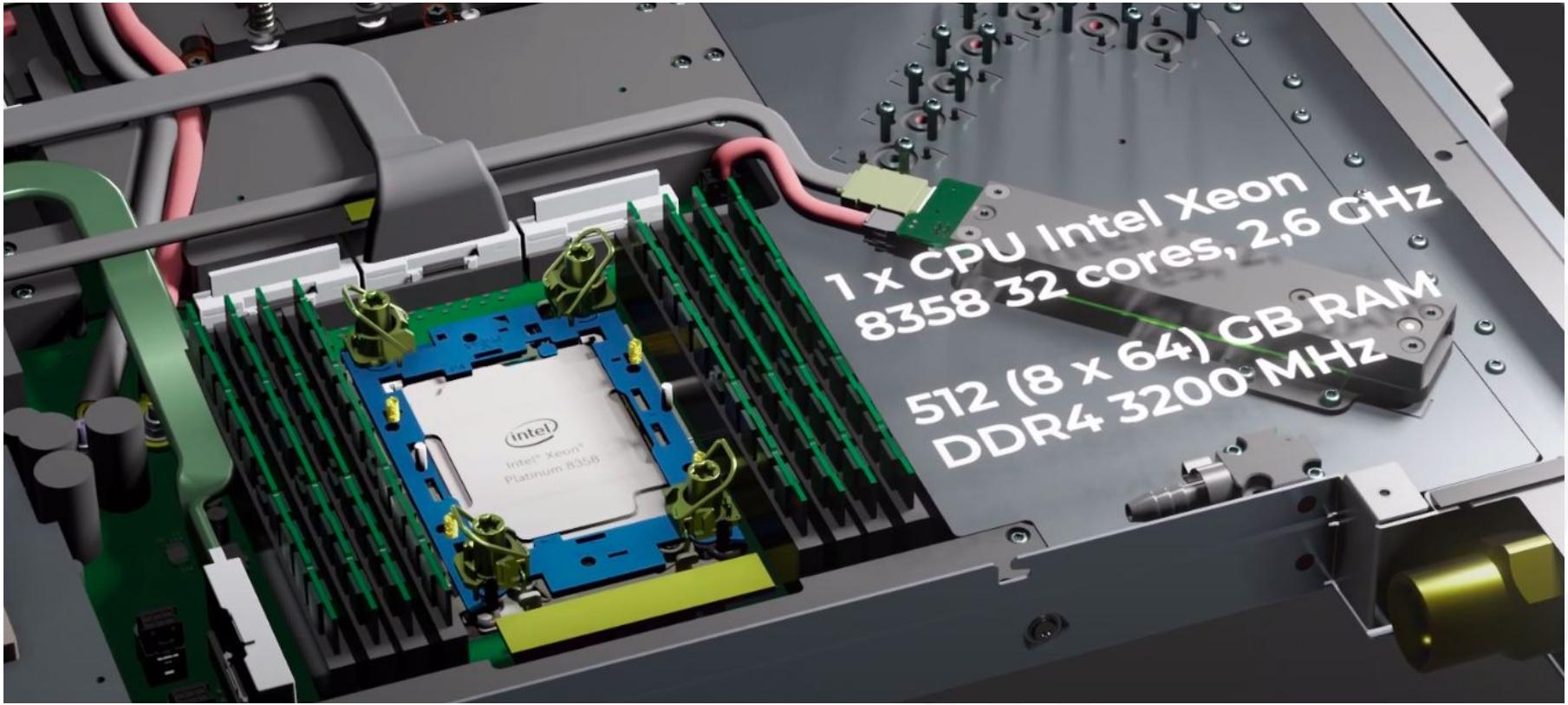


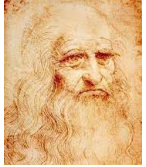
LEONARDO: Booster Module



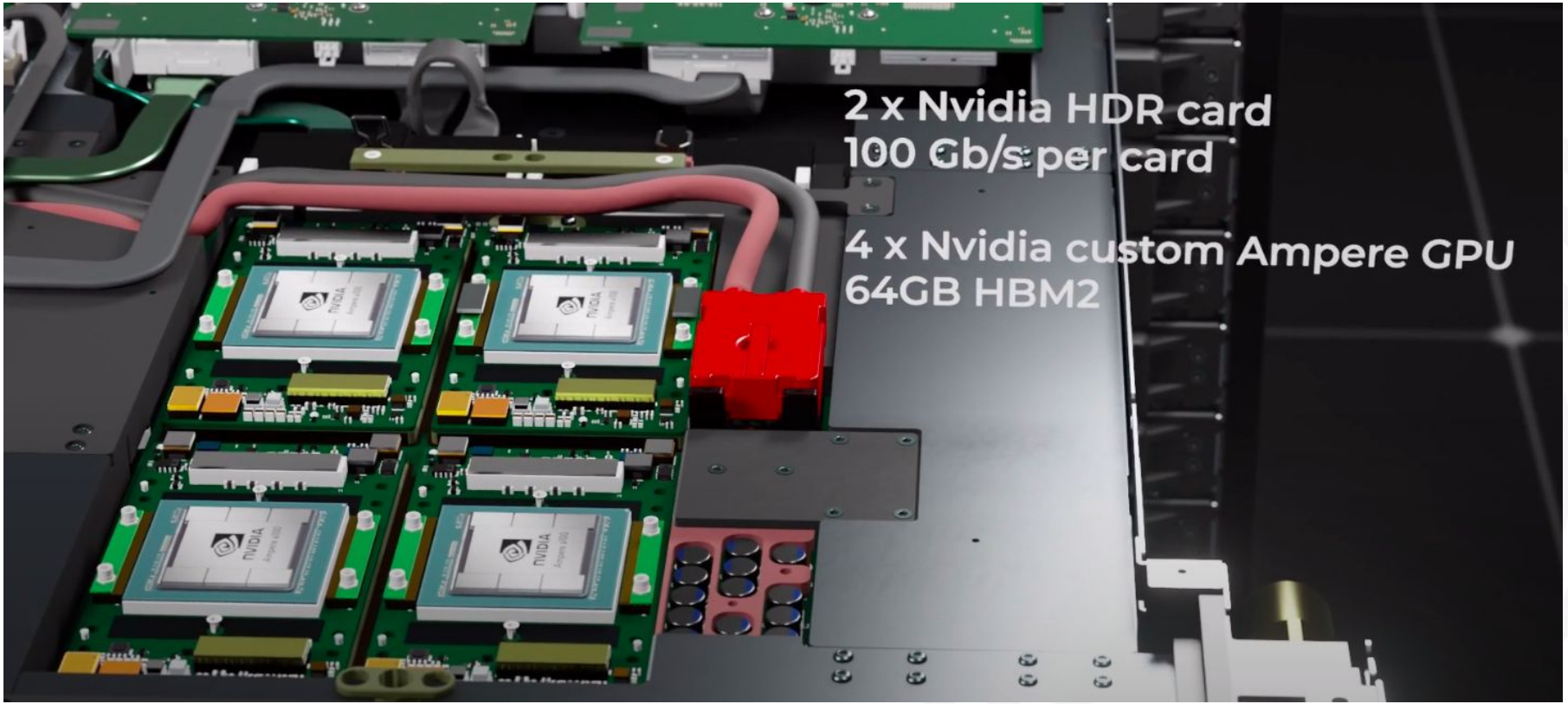


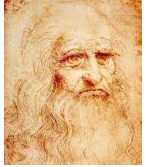
LEONARDO: Booster Module



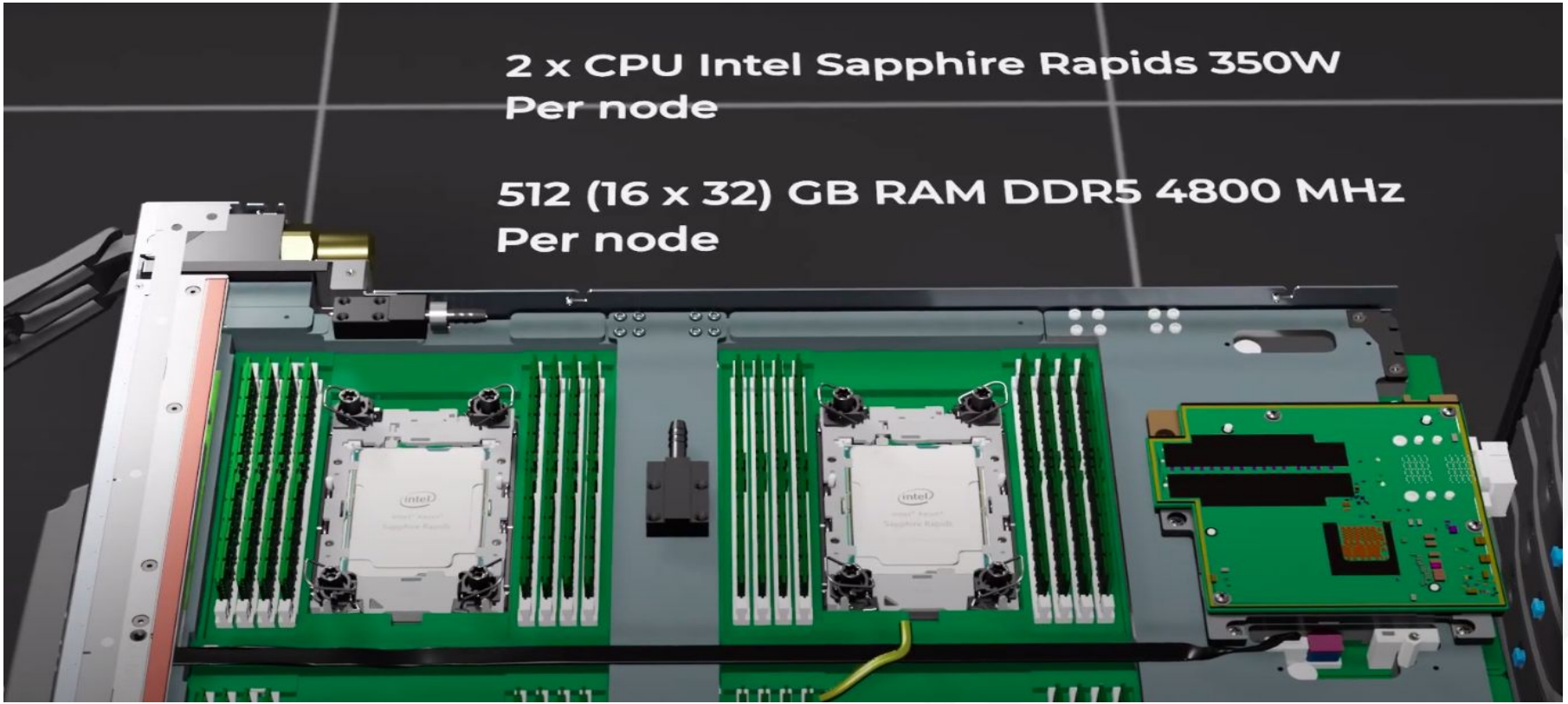


LEONARDO: Booster Module





LEONARDO: Data Centric Module



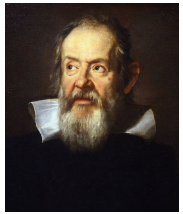
CINECA HPC Infrastructure

LEONARDO

At the moment the **4th** Supercomputer in the world. It will be available from August 2023.
~5000 compute nodes from 2 partitions: **Booster** and **Data centric** modules.
3456 Booster compute nodes: CPU Intel Xeon 8358 "**IceLake**" with **4x Nvidia Ampere. 309.6 Pflop/s** of total computing power

MARCONI

The actual configuration consists of Marconi-A3 with 3188 nodes **Intel SkyLake** (in production since August 2017). Marconi was classified in the top500, rank 12 in November 2016.



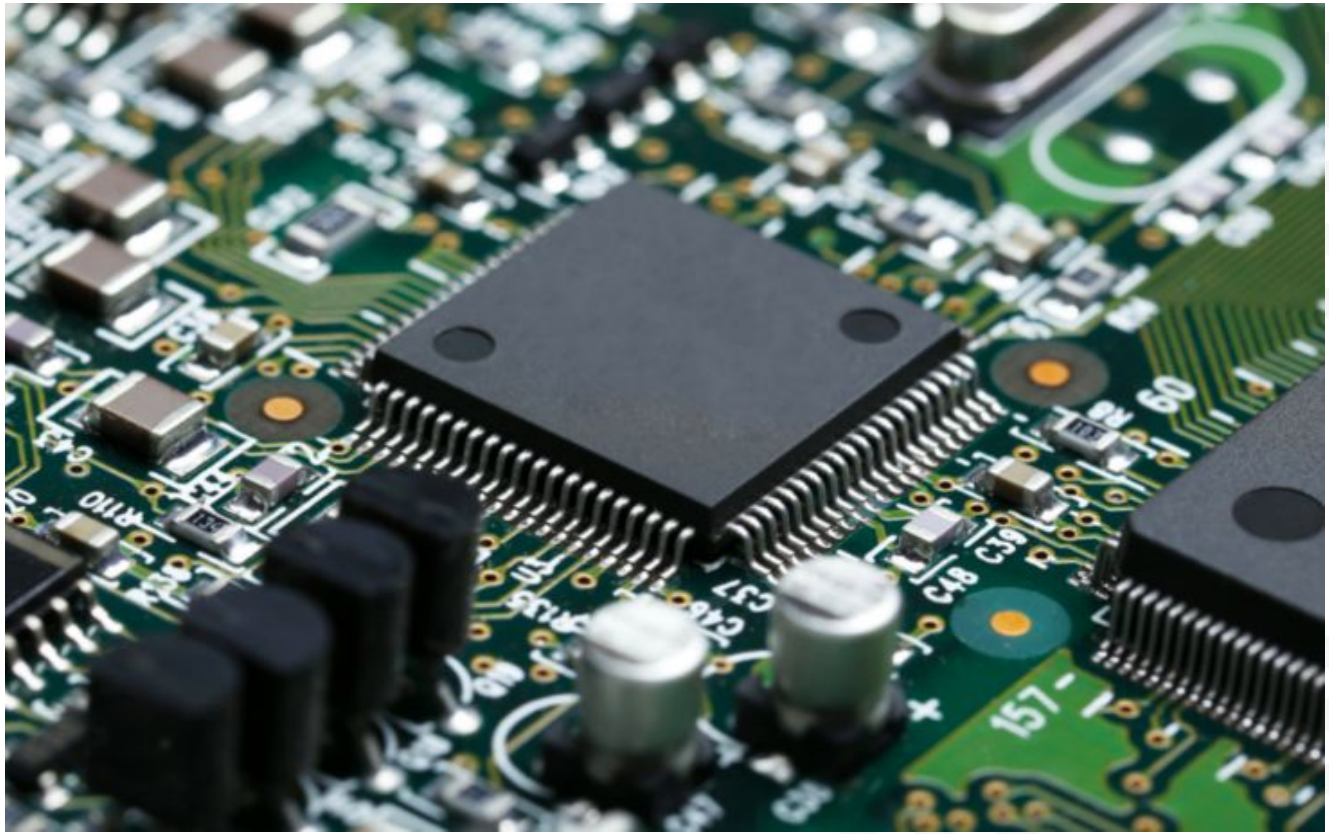
GALILEO100 aka G100

From September 2021 and co-funded by the European ICEI (Interactive Computing e-Infrastructure) project.
528 computing nodes with **Intel CascadeLake** and 36 nodes with 2x **NVIDIA GPU V100** with 100Gbs Infiniband interconnection.

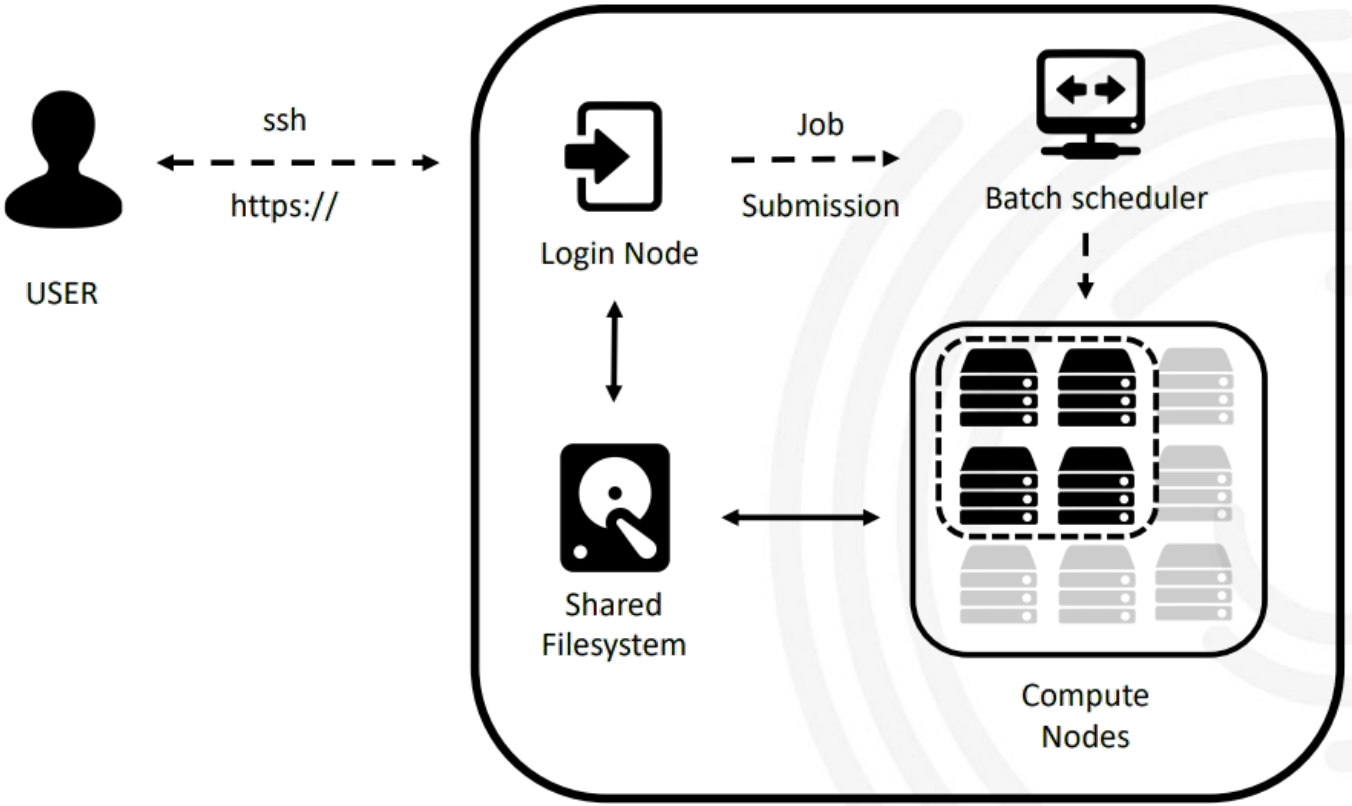
CLOUD HPC

CINECA cloud service has been renewed in April 2020 with Intel E5-2697 v4 (Broadwell) nodes, completing the HPC ecosystem.

CINECA Architectures



How-To Use a Cluster @CINECA



Login Nodes

When you log in Leonardo, you find yourself in one of the four login nodes, selected in round robin fashion to balance out the load of users.

Interactive runs on login nodes are **strongly discouraged** and should be limited to short test runs.

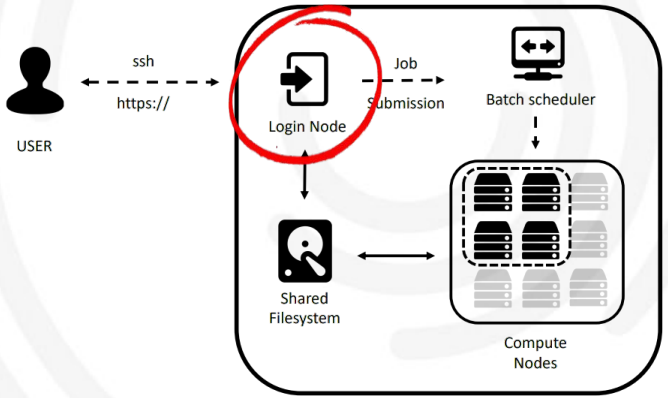
There are per user limits on cpu-time (10 minutes) and memory (1 GB) **IMPORTANT**: avoid running large parallel applications on the front-ends!

Leonardo is a general purpose system used by hundreds of users.

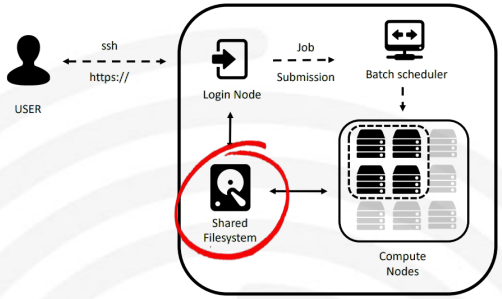
How-To Access to Leonardo:

```
$ ssh <USER>@login.leonardo.cineca.it
```

- `$ ssh <USER>@login01-ext.leonardo.cineca.it`
- `$ ssh <USER>@login02-ext.leonardo.cineca.it`
- `$ ssh <USER>@login05-ext.leonardo.cineca.it`
- `$ ssh <USER>@login07-ext.leonardo.cineca.it`



Work Areas @CINECA Clusters



$HOME

Permanent, backed-up, and local to Leonardo. 50 Gb of quota.
For source code or important input files.

$CINECA_SCRATCH

Large, parallel file-system. No quota. Run your simulations and calculations here.
A cleaning policy will delete all your files older than 40 days.

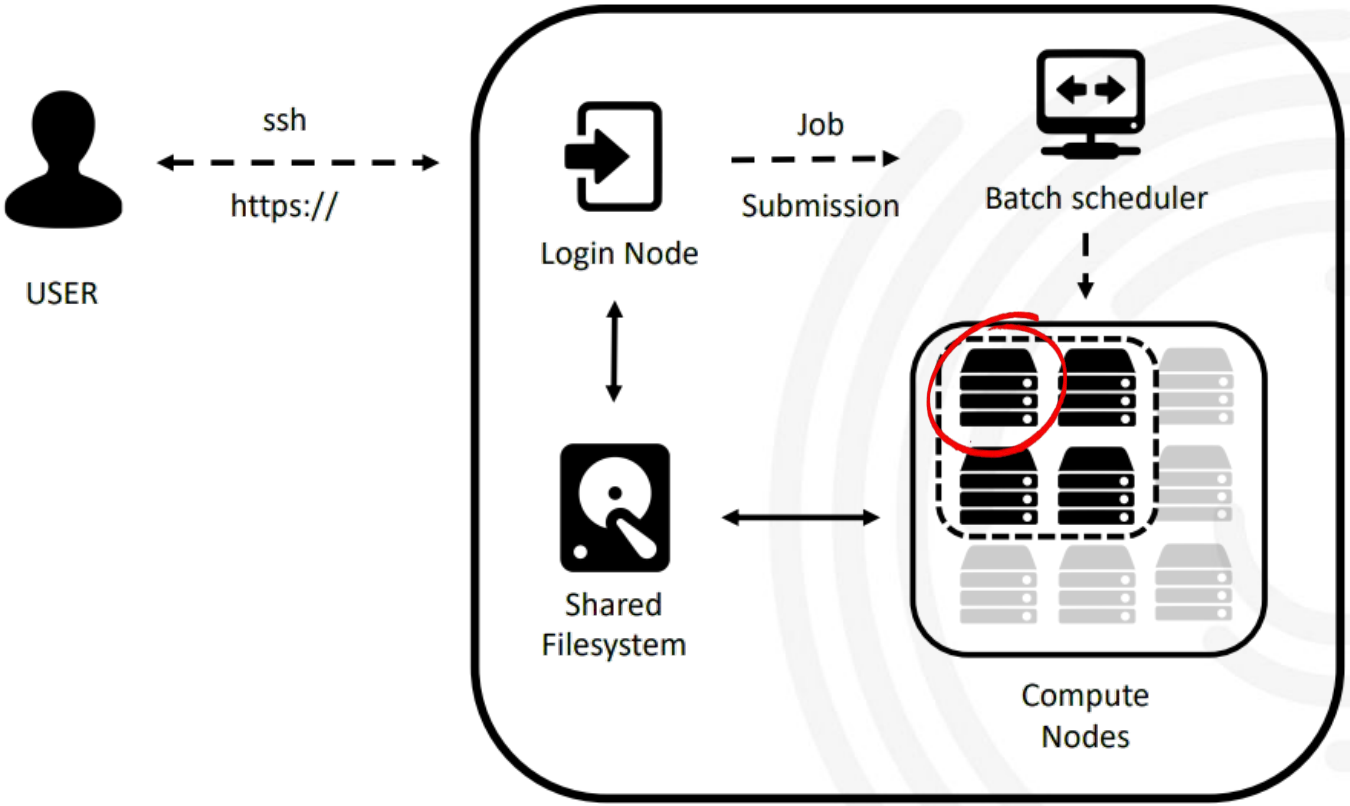
$WORK

Similar to `$CINECA_SCRATCH`, but the content is shared among all the users of the same account. 1 Tb of quota (no cleaning policy).

Use the command **`cindata`** to get info on your disk occupation

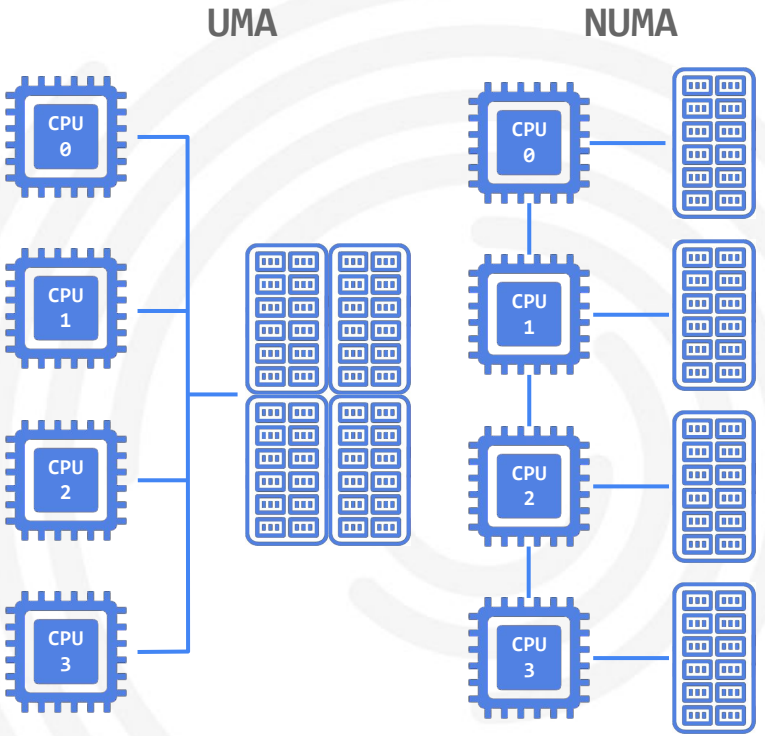
https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.5%3A+Data+storage+and+FileSystems

CINECA Architectures



Parallel Computers Memory Systems

- *Uniform Memory Access* (**UMA**)
  - All processors share the physical memory **uniformly**.
  - Access time to a memory location is independent of which processor makes the request.
- *Non-Uniform Memory Access* (**NUMA**)
  - Starting from AMD Opteron (2003) and Intel Nehalem (2007) the CPU memory controller was integrated directly into the CPU improving memory performance.
  - Each CPU gets its **own pool of memory** bandwidth.
  - Any CPU can access any memory location, but accesses are much faster to memory "**local**" (physical address directly attached) to CPU than "**non-local**" accesses.
  - Non-local accesses are resolved by using HyperTransport (HT) / QuickPath Interconnect (QPI).
  - Cache coherence problem.



Binding & Affinity

It is of crucial important to take into account the **structure** of the **system architecture** in order to achieve an efficient parallel program.

Special attention should be given to ensuring that all memory accesses are **local**.

Therefore, processes/threads should be binded to a central processing unit (CPU) or to a range of CPUs, so that the process/thread will execute only on the designated CPU rather than any CPU.

Binding means that a process/thread will be dispatched to a given processor only.

This is also enabled through setting CPU "thread **affinity**" so the operating system schedules threads onto the right CPUs.

Processor ***affinity*** is the probability of dispatching of a thread to the processor that was previously executing it. NB: If a thread is interrupted and then redispatched to the same processor, the caches might still contain lines that belong to the thread.

NB: process/thread binding can be useful for CPU-intensive programs with few interrupts.

Graphics Processing Unit Coming

Graphics Processing Unit (GPU) is a device equipped with **highly parallel microprocessor** (thousands of cores) and a private memory with very **high bandwidth**. GPU highly parallel structure makes them more efficient than CPUs for embarrassingly parallel algorithms.

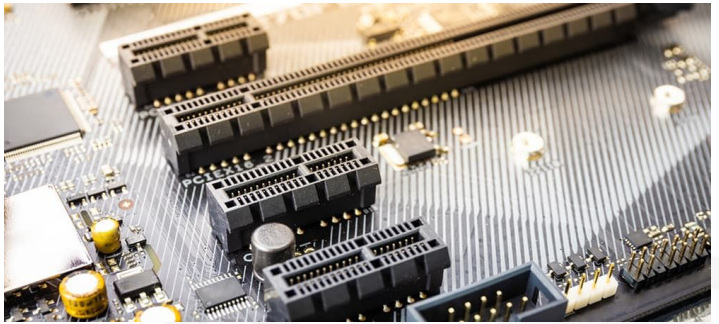
- Born in 1990s as a response to the growing demand for high definition **3D rendering graphic** applications (gaming, animations, etc).
- The increasing popularity of 3D-accelerated games caused a rapid growth of computational capabilities of modern GPUs.
- in 2000s manufacturers extended the GPUs' core functionality with the introduction of **pixel and vertex** shader languages.
- Brave computer scientists used the increasing computational power of GPUs to implement more general algorithms by expressing them in shader languages. This was the birth of the so-called general-purpose computing on GPUs (**GPGPU**).
- In 2006 NVIDIA released the Compute Unified Device Architecture (**CUDA**) extension to C/C++ for GPU architecture

CPU vs GPU

A Central Processing Unit (CPU) is a latency-optimized general purpose processor designed to handle a wide range of tasks sequentially, while a Graphics Processing Unit (GPU) is a throughput-optimized specialized processor designed for high-end parallel computing.

- **Massive Parallel Computing**: extensive calculations with similar operations
- **High Data Throughput**: thousands of cores performing the same operation on multiple data items in parallel
- **High Computing Throughput**: high-performance computing power

Host/Device Data Movements

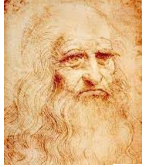


Available Connection Technologies:

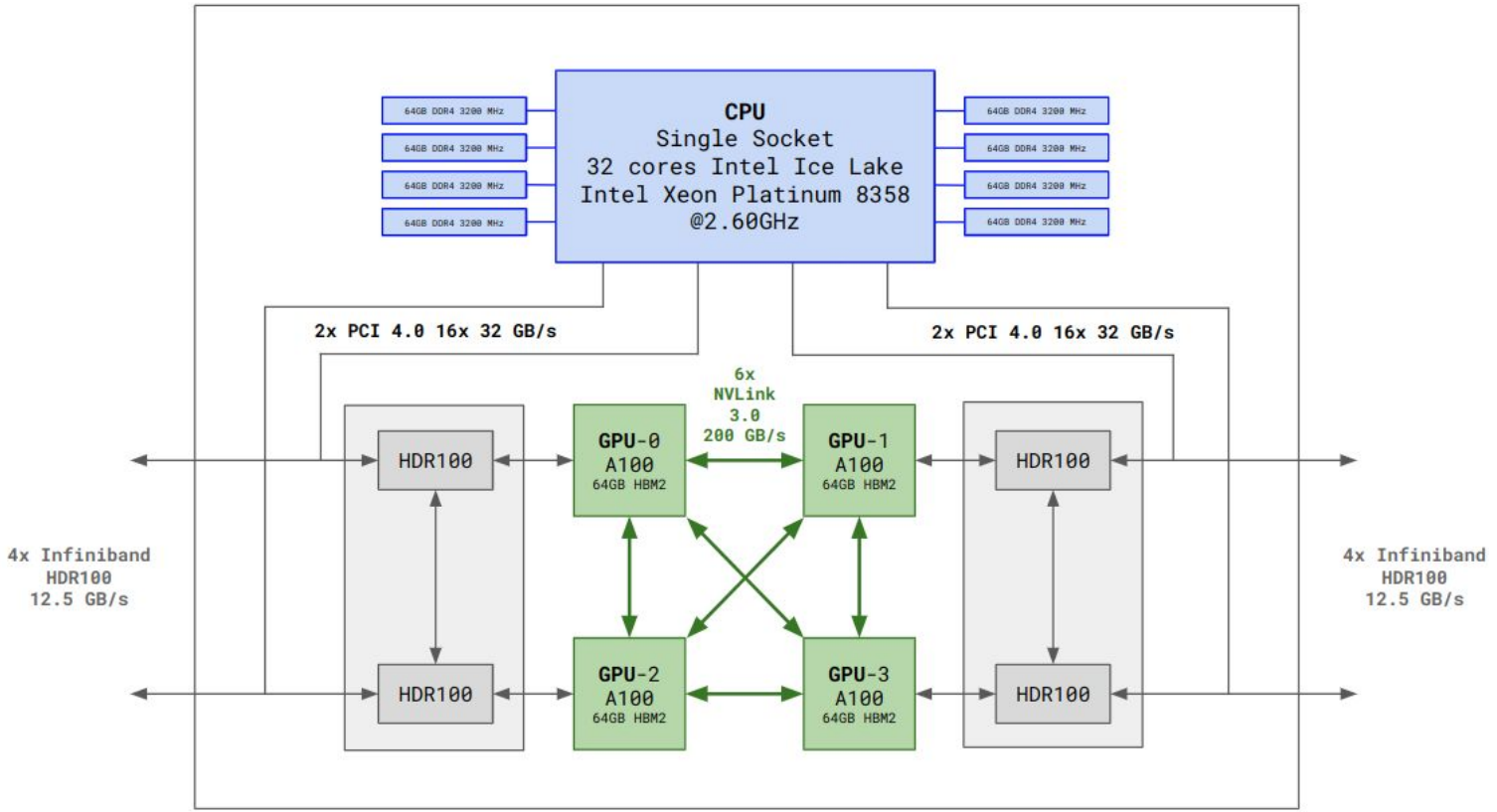
- **PCI Express** link
- **NVLink** nVIDIA
- **Infinity Fabric** AMD

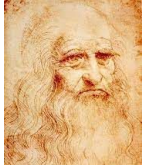
NVLink performance				
Version	Year	TR GT/s	Lane	Tot BW GB/s
1.0	2016	20	x4	80
2.0	2017	25	x6	150
3.0	2020	25/50	x6	300

PCI Express link performance		
Version	Year	BW x16 GB/s
1.0	2003	4.0
2.0	2007	8.0
3.0	2010	15.7
4.0	2017	31.5
5.0	2019	63.0



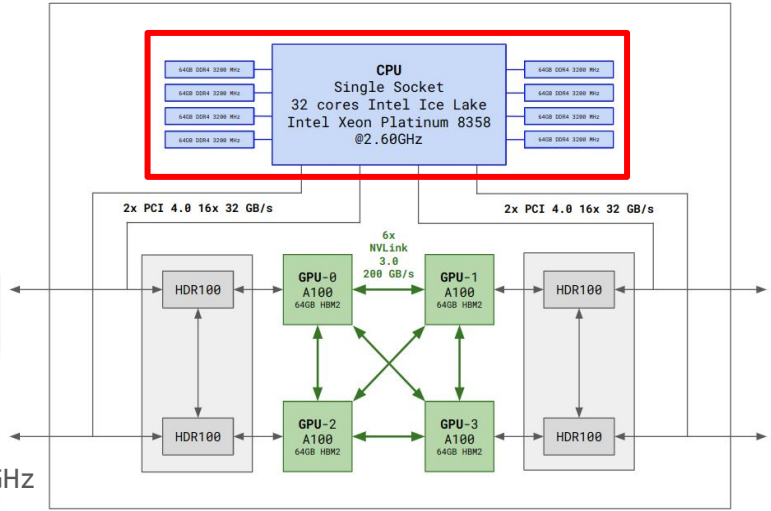
LEONARDO: Booster Module

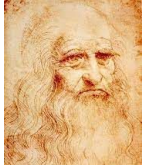




CPU Architecture

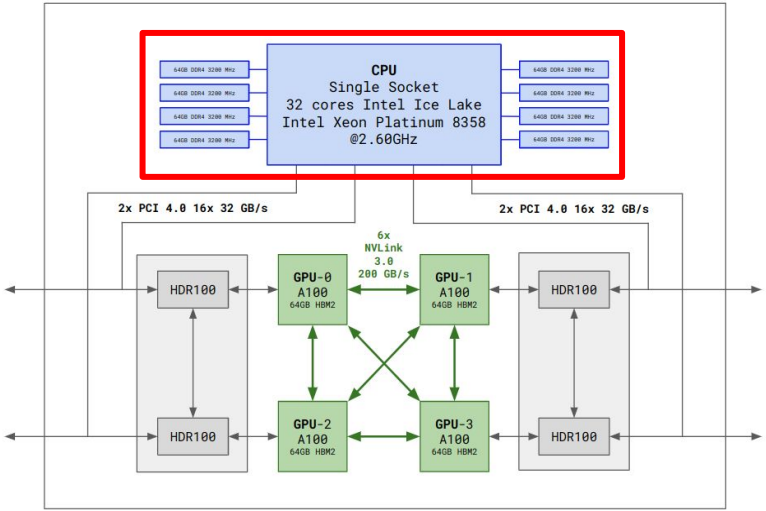
```
$ lscpu

Architecture:          x86_64
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    1
Core(s) per socket:    32
Socket(s):             1
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 106
Model name:            Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz
CPU MHz:               3338.495
CPU max MHz:           2601,0000
CPU min MHz:           800,0000
L1d cache:             48K
L1i cache:             32K
L2 cache:              1280K
L3 cache:              49152K
NUMA node0 CPU(s):     0-15
NUMA node1 CPU(s):     16-31
```

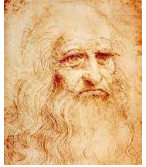




CPU Architecture

```
$ numactl -H

available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 256924 MB
node 0 free: 247406 MB
node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
node 1 size: 257999 MB
node 1 free: 254070 MB
node distances:
node   0   1
  0:  10  11
  1:  11  10
```





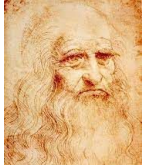
GPU Nvidia Ampere

The NVIDIA System Management Interface (aka **`nvidia-smi`**) is a command for monitoring NVIDIA GPU devices.

Several details are listed such as:

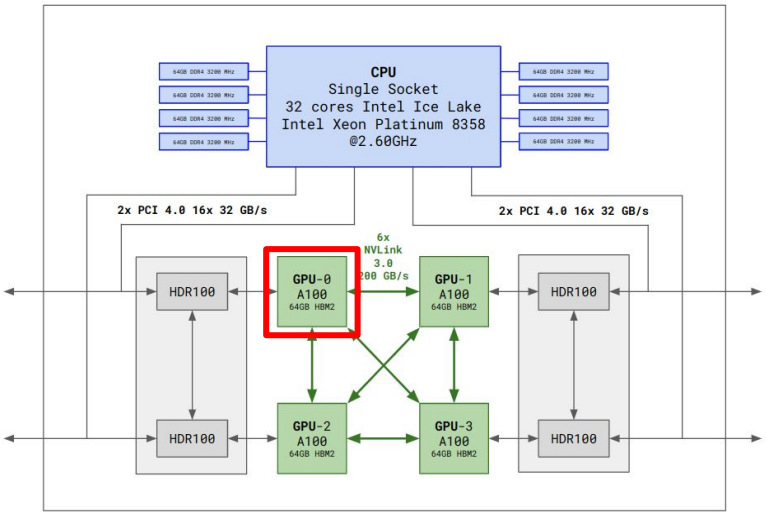
- the CUDA and GPU driver versions,
- the number and type of GPUs available,
- the GPU memory each,
- running GPU process,
- etc.

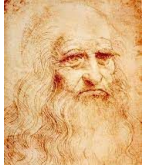
NB: with the option `-l SEC`, `nvidia-smi` continuously reports query data at the specified interval in seconds. Useful for monitoring resource utilisation on a node.



GPU Nvidia Ampere

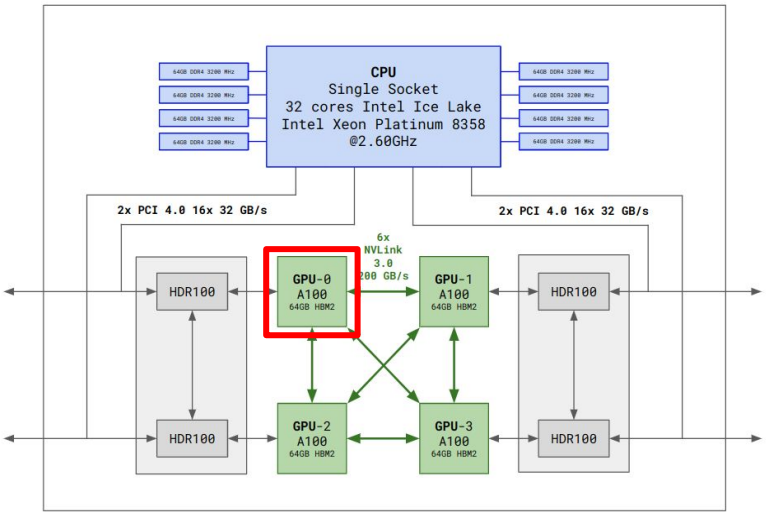
```
$ nvidia-smi

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 520.61.05    Driver Version: 520.61.05    CUDA Version: 11.8     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA PG506-243    On   | 00000000:1D:00.0 Off |                    0 |
| N/A   38C    P0    60W / 467W |      0MiB / 65536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA PG506-243    On   | 00000000:56:00.0 Off |                    0 |
| N/A   39C    P0    59W / 456W |      0MiB / 65536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA PG506-242    On   | 00000000:8F:00.0 Off |                    0 |
| N/A   38C    P0    61W / 469W |      0MiB / 65536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA PG506-242    On   | 00000000:C8:00.0 Off |                    0 |
| N/A   39C    P0    61W / 461W |      0MiB / 65536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
```

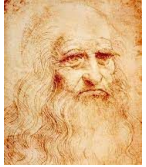




GPU Nvidia Ampere

```
$ DeviceQuery

Device 0: "NVIDIA PG506-243"
  CUDA Driver Version / Runtime Version          11.8 / 11.8
  CUDA Capability Major/Minor version number:    8.0
  Total amount of global memory:                 64969 MBytes (68125065216 bytes)
  (124) Multiprocessors, (064) CUDA Cores/MP:    7936 CUDA Cores
  GPU Max Clock rate:                            1395 MHz (1.39 GHz)
  Memory Clock rate:                             1593 Mhz
  Memory Bus Width:                              4096-bit
  L2 Cache Size:                                 33554432 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 163
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        167936 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 5 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 29 / 0
```





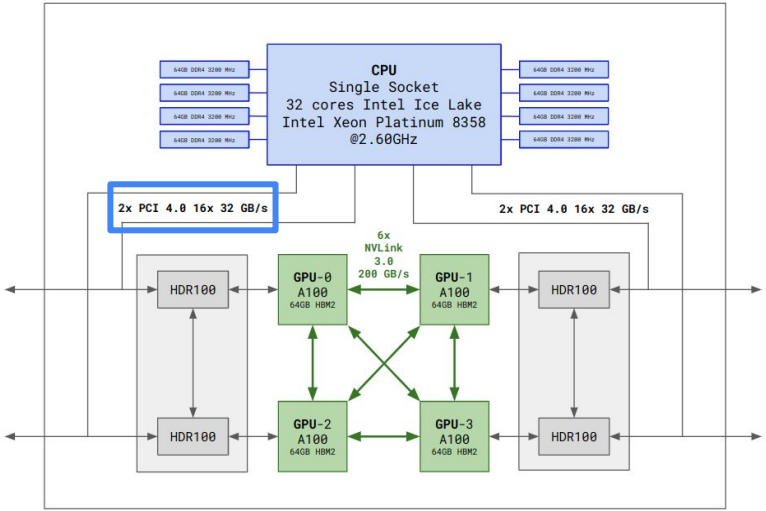
Communication Buses

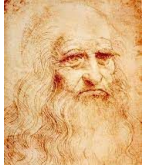
PCI Express 4th Gen.

- 4x CPU/GPU Connection
- **32 GB/s** Bidirectional Bandwidth (16x lanes)

NVLink 3.0

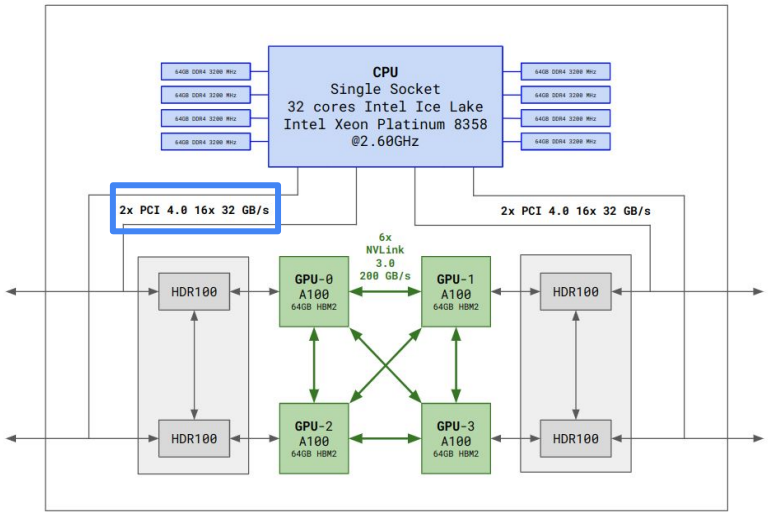
- NVIDIA high-speed coherent interconnect Technology GPU-to-GPU and GPU-to-CPU
- 6x GPU/GPU Connection
- **200 GB/s** peak Bidirectional Bandwidth

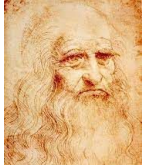




CPU-GPU Communication with PCI Exp.

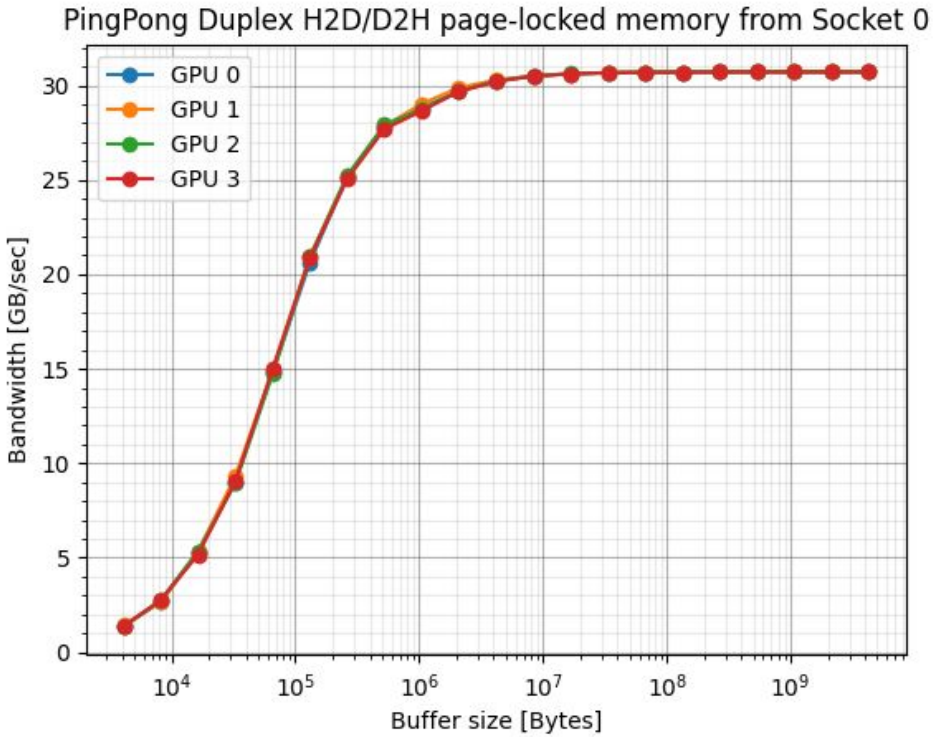
```
$ bandwidthTest

[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: NVIDIA PG506-243
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(GB/s)
   32000000                 24.9

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(GB/s)
   32000000                 25.9

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(GB/s)
   32000000                 1163.9

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

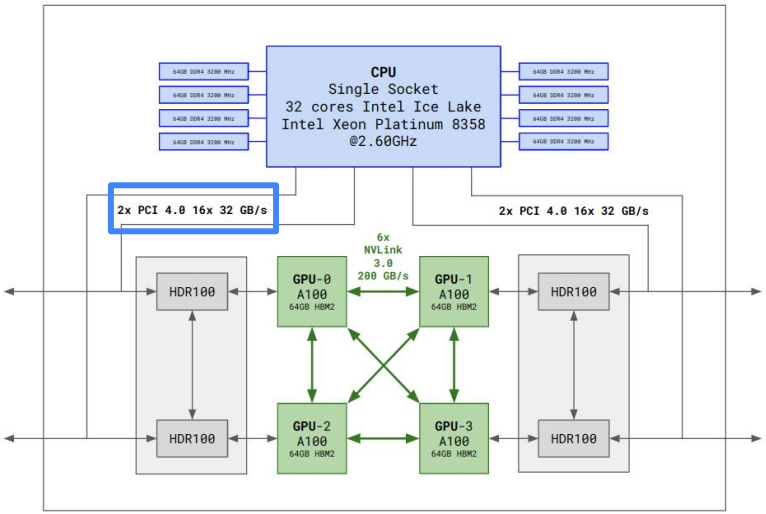


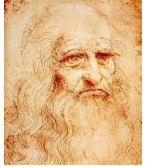


CPU-GPU Communication with PCI Exp.

PingPong Duplex mode H2D/D2H Page-locked Memory







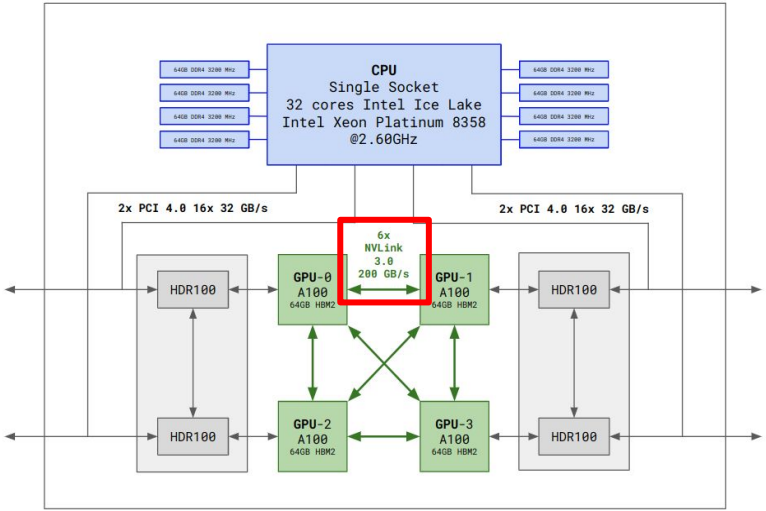
Communication Buses

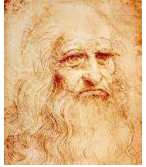
PCI Express 4th Gen.

- 4x CPU/GPU Connection
- **32 GB/s** Bidirectional Bandwidth (16x lanes)

NVLink 3.0

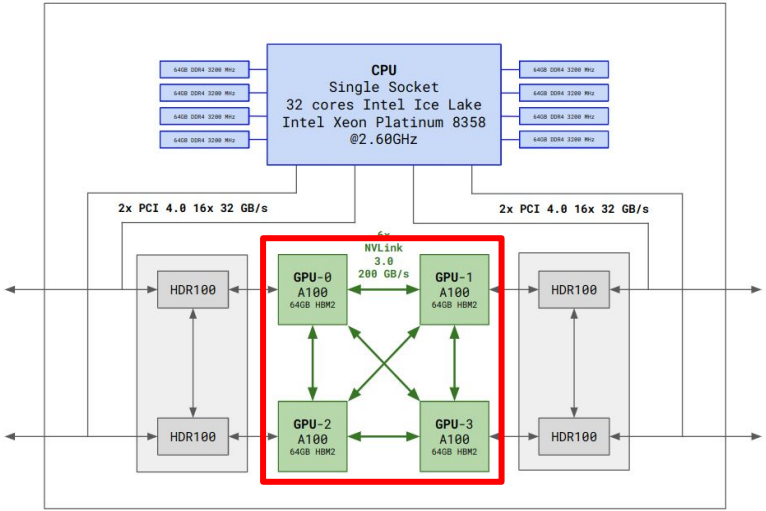
- NVIDIA high-speed coherent interconnect Technology GPU-to-GPU and GPU-to-CPU
- 6x GPU/GPU Connection
- **200 GB/s** peak Bidirectional Bandwidth

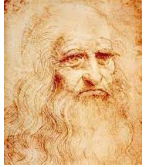




GPU Topology

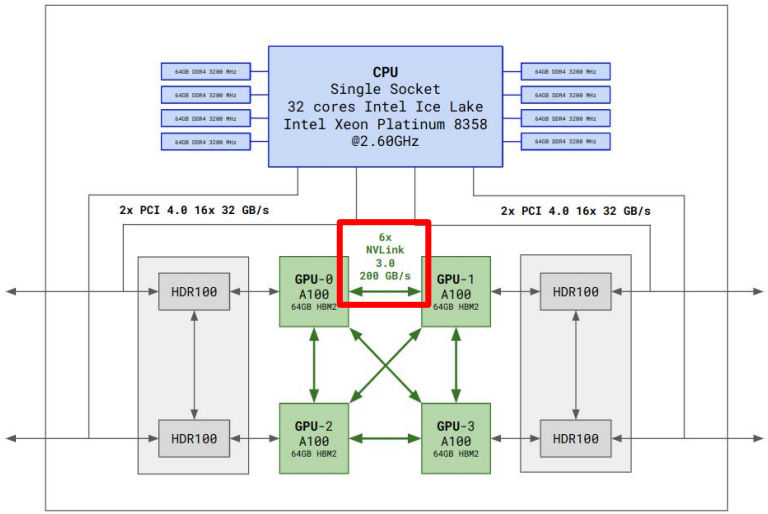
```
$ nvidia-smi topo -m
      GPU0  GPU1  GPU2  GPU3  mlx5_0      mlx5_1      mlx5_2
GPU0   X    NV4   NV4   NV4   PXB   SYS   SYS   SYS   0     0-1
GPU1  NV4    X    NV4   NV4   SYS   PXB   SYS   SYS   0     0-1
GPU2  NV4   NV4    X    NV4   SYS   SYS   PXB   SYS   0     0-1
GPU3  NV4   NV4   NV4    X    SYS   SYS   SYS   PXB   0     0-1
mlx5_0      PXB   SYS   SYS   SYS    X    SYS   SYS   SYS
mlx5_1      SYS   PXB   SYS   SYS   SYS    X    SYS   SYS
mlx5_2      SYS   SYS   PXB   SYS   SYS   SYS    X    SYS
mlx5_3      SYS   SYS   SYS   PXB   SYS   SYS   SYS    X

Legend:

  X    = Self
  SYS  = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
  NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA
node
  PHB  = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
  PXB  = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
  PIX  = Connection traversing at most a single PCIe bridge
  NV#  = Connection traversing a bonded set of # NVLinks
```

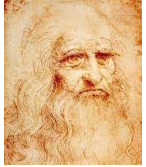




GPU-GPU Communication with NVLink

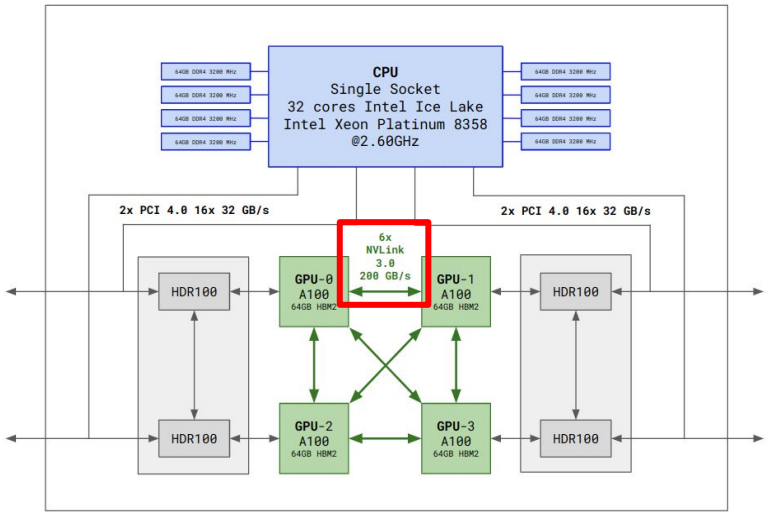
```
$ nvidia-smi nvlink --status -i 0

GPU 0: NVIDIA PG506-243 (UUID: GPU-1ca9dfcd-74cb-a877-70c5-be146530ac8f)
Link 0: 25 GB/s
Link 1: 25 GB/s
Link 2: 25 GB/s
Link 3: 25 GB/s
Link 4: 25 GB/s
Link 5: 25 GB/s
Link 6: 25 GB/s
Link 7: 25 GB/s
Link 8: 25 GB/s
Link 9: 25 GB/s
Link 10: 25 GB/s
Link 11: 25 GB/s
```

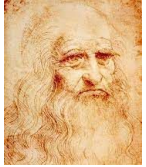




GPU-GPU Communication with NVLink

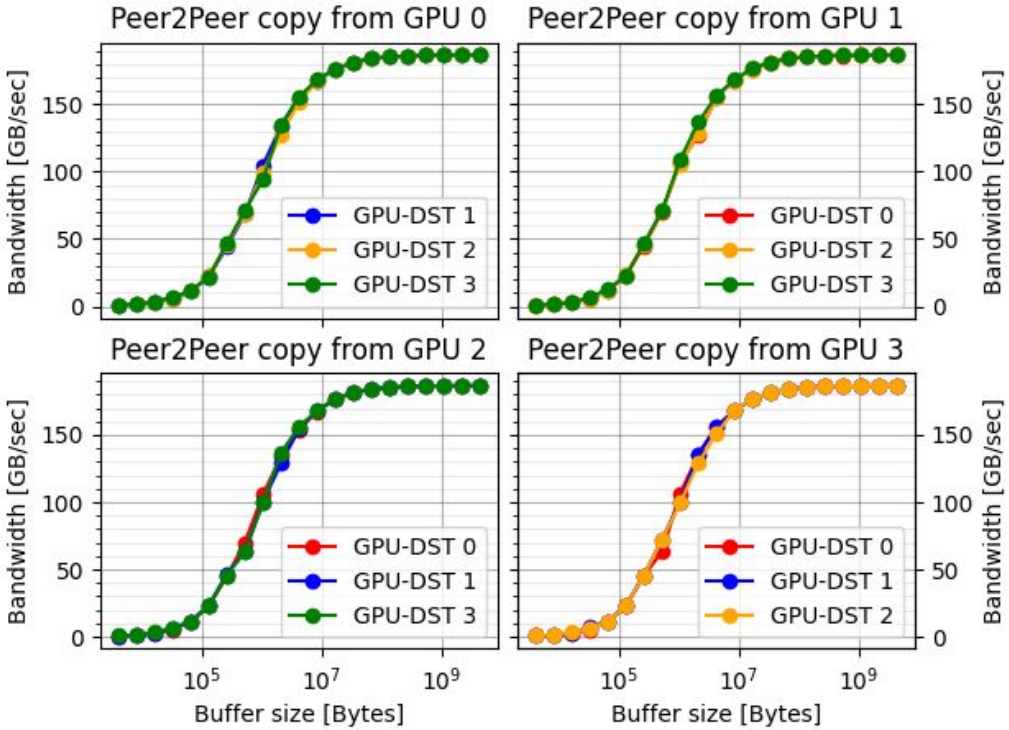
```
$ p2pBandwidthLatencyTest

Unidirectional P2P=Enabled Bandwidth (P2P Writes) Matrix (GB/s)
  D\D     0       1       2       3
    0 1286.01   93.55   93.60   93.39
    1   93.57 1335.47   93.35   93.48
    2   93.55   93.37 1335.47   93.61
    3   93.29   93.56   93.55 1337.76

Bidirectional P2P=Enabled Bandwidth Matrix (GB/s)
  D\D     0       1       2       3
    0 1309.72  185.13  185.35  184.87
    1  185.40 1344.66  185.00  185.46
    2  185.36  184.96 1345.82  185.33
    3  185.11  185.48  185.28 1348.14

P2P=Enabled Latency (P2P Writes) Matrix (us)
  GPU     0       1       2       3
    0    2.37    2.24    2.27    2.25
    1    2.30    2.28    2.23    2.19
    2    2.27    2.22    2.38    2.26
    3    2.19    2.27    2.28    2.52
```

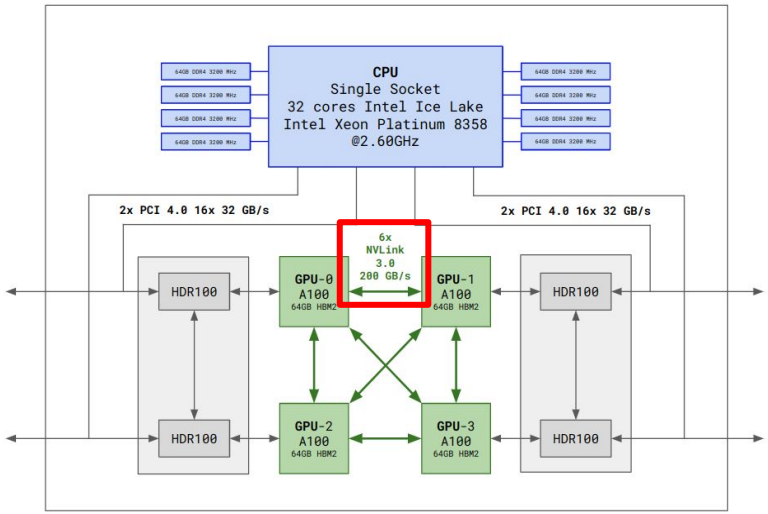


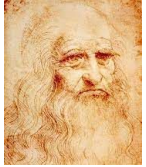


GPU-GPU Communication with NVLink

PingPong Peer2Peer copies - P2P Enabled

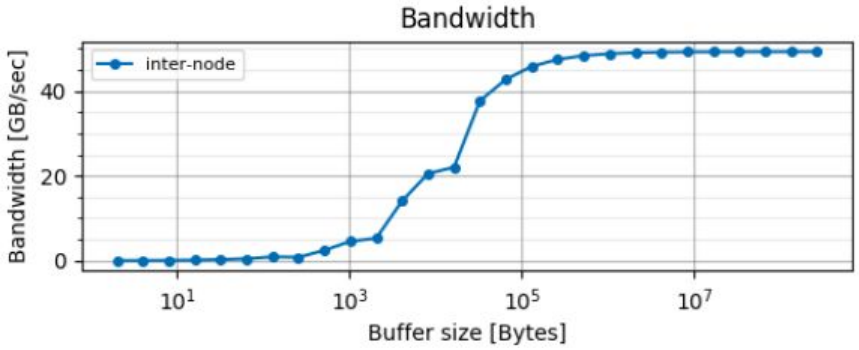




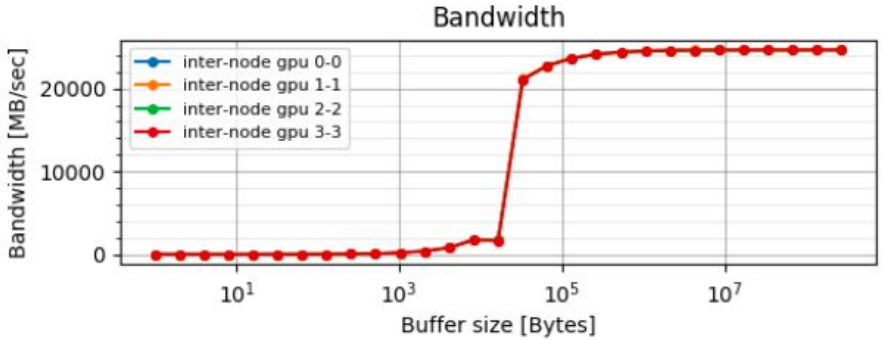


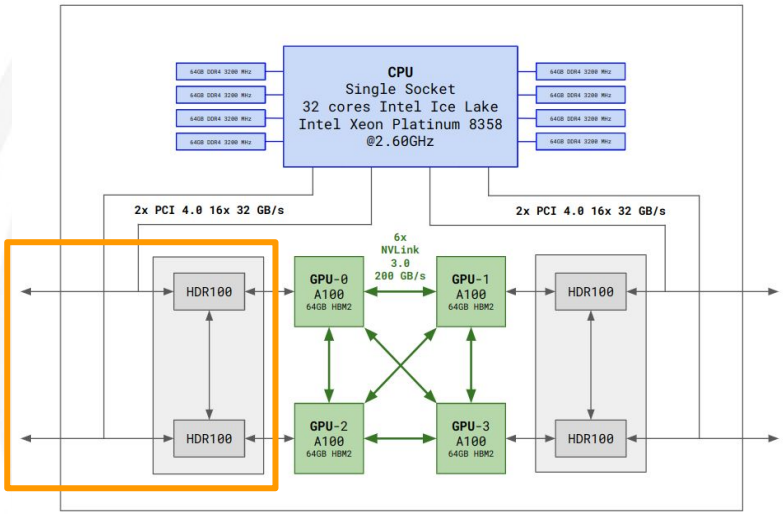
Inter-Node Commun. with MPI-CUDA-AWARE

- MPI Inter-node pt2pt Bidirectional Device-Device



- MPI Inter-node pt2pt Bidirectional Device-Device





Production Environment

The Module Software Environment

At the first login you will find already several common **softwares** installed.

The tools to handle with these software are provided by the **module environment**. It is automatically loaded at the login.

The available softwares are divided in profiles and categories for an easier searching.

PROFILES	
`base`	Contains basic modules for programming and code development

`astro`
`bioinf`
`chem-phys`
`deeplrn`
`lifesc`
	Software specific for a research community

`advanced`
`archive`
`candidate`
	Additional softwares

CATEGORIES

`compilers`
`environment`
`libraries`
`tools`
`data`
`applications`

The Module Software Environment

Available modules can be shown with the command: *module available*

```
$ module av

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/profiles ------------------------------------------------------------------------
profile/archive  profile/base  profile/candidate  profile/chem-phys  profile/deeplrn  profile/lifesc  profile/meteo  profile/spoke7

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/libraries ------------------------------------------------------------------------
adios/1.13.1--openmpi--4.1.4--gcc--11.3.0              hdf5/1.12.2--gcc--11.3.0-threadsafe                netlib-scalapack/2.2.0--openmpi--4.1.4--nvhpc--23.1
adios/1.13.1--openmpi--4.1.4--nvhpc--23.1              hdf5/1.12.2--openmpi--4.1.4--gcc--11.3.0           netlib-xblas/1.0.248--gcc--11.3.0
blitz/1.0.2--gcc--11.3.0                               hdf5/1.12.2--openmpi--4.1.4--nvhpc--23.1           openblas/0.3.21--gcc--11.3.0
boost/1.80.0--gcc--11.3.0                              intel-oneapi-mkl/2022.2.1                          openblas/0.3.21--nvhpc--23.1
boost/1.80.0--openmpi--4.1.4--gcc--11.3.0              intel-oneapi-mpi/2021.7.1                          openmpi/4.1.4--gcc--11.3.0-cuda-11.8
boost/1.80.0--openmpi--4.1.4--nvhpc--23.1              intel-oneapi-tbb/2021.7.1                          openmpi/4.1.4--nvhpc--23.1-cuda-11.8
cgal/5.4.1--gcc--11.3.0                                libmatheval/1.1.11--gcc--11.3.0                    parallel-netcdf/1.12.3--openmpi--4.1.4--gcc--11.3.0
cgal/5.4.1--openmpi--4.1.4--gcc--11.3.0                libszip/2.1.1--gcc--11.3.0                         parallel-netcdf/1.12.3--openmpi--4.1.4--nvhpc--23.1
cineca-hpyc/2023.05                                    magma/2.6.2--gcc--11.3.0-cuda-11.8                 parmetis/4.0.3--openmpi--4.1.4--gcc--11.3.0
cudnn/8.4.0.27-11.6--gcc--11.3.0                       metis/5.1.0--gcc--11.3.0                           parmetis/4.0.3--openmpi--4.1.4--nvhpc--23.1
cutensor/1.5.0.3--gcc--11.3.0                          nccl/2.14.3-1--gcc--11.3.0-cuda-11.8               petsc/3.18.1--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
elpa/2021.11.001--openmpi--4.1.4--gcc--11.3.0-cuda-11.8  netcdf-c/4.9.0--gcc--11.3.0                     petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-complex
fftw/3.3.10--gcc--11.3.0                               netcdf-c/4.9.0--openmpi--4.1.4--gcc--11.3.0        petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--gcc--11.3.0               netcdf-c/4.9.0--openmpi--4.1.4--nvhpc--23.1        petsc/3.19.0--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--nvhpc--23.1               netcdf-fortran/4.6.0--gcc--11.3.0                  proj/8.2.1--gcc--11.3.0
gdal/3.5.3--gcc--11.3.0                                netcdf-fortran/4.6.0--openmpi--4.1.4--gcc--11.3.0  rapids/2023.09
gsl/2.7.1--gcc--11.3.0                                 netcdf-fortran/4.6.0--openmpi--4.1.4--nvhpc--23.1  slate/2022.07.00--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
hdf5/1.12.2--gcc--11.3.0                               netlib-scalapack/2.2.0--openmpi--4.1.4--gcc--11.3.0  zlib/1.2.13--gcc--11.3.0

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/tools ------------------------------------------------------------------------
anaconda3/2022.05  emacs/28.2             gnuplot/5.4.3--gcc--11.3.0   nco/5.0.1--openmpi--4.1.4--gcc--11.3.0     singularity/3.8.7  texinfo/6.5
anaconda3/2023.03  git-lfs/3.1.2          intel-oneapi-vtune/2022.4.1  ncview/2.1.8--openmpi--4.1.4--gcc--11.3.0  snakemake/6.15.1   texinfo/6.5--gcc--11.3.0
cmake/3.24.3       git/2.38.1             jube/2.4.3                   ninja/1.11.1                               spack/0.19.1-d71   valgrind/3.19.0--openmpi--4.1.4--gcc--11.3.0
curl/7.79.0        git/2.38.1--nvhpc--23.1  maven/3.8.4                openjdk/11.0.17_8                          superc/2.0

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/compilers ------------------------------------------------------------------------
cuda/11.8   intel-oneapi-compilers/2023.0.0     nvhpc/22.3  perl/5.36.0--gcc--8.5.0   perl/5.36.0--nvhpc--23.1   python/3.10.8--gcc--11.3.0
gcc/11.3.0  llvm/15.0.4--gcc--11.3.0-cuda-11.8  nvhpc/23.1  perl/5.36.0--gcc--11.3.0  python/3.10.8--gcc--8.5.0
```

The Module Software Environment

Available modules can be shown with the command: *module available*

```
$ module av

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/profiles -------------------------------------------------------------------------
profile/archive  profile/base  profile/candidate  profile/chem-phys  profile/deeplrn  profile/lifesc  profile/meteo  profile/spoke7

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/libraries -------------------------------------------------------------------------
adios/1.13.1--openmpi--4.1.4--gcc--11.3.0               hdf5/1.12.2--gcc--11.3.0-threadsafe                 netlib-scalapack/2.2.0--openmpi--4.1.4--nvhpc--23.1
adios/1.13.1--openmpi--4.1.4--nvhpc--23.1               hdf5/1.12.2--openmpi--4.1.4--gcc--11.3.0            netlib-xblas/1.0.248--gcc--11.3.0
blitz/1.0.2--gcc--11.3.0                                hdf5/1.12.2--openmpi--4.1.4--nvhpc--23.1            openblas/0.3.21--gcc--11.3.0
boost/1.80.0--gcc--11.3.0                               intel-oneapi-mkl/2022.2.1                           openblas/0.3.21--nvhpc--23.1
boost/1.80.0--openmpi--4.1.4--gcc--11.3.0               intel-oneapi-mpi/2021.7.1                           openmpi/4.1.4--gcc--11.3.0-cuda-11.8
boost/1.80.0--openmpi--4.1.4--nvhpc--23.1               intel-oneapi-tbb/2021.7.1                           openmpi/4.1.4--nvhpc--23.1-cuda-11.8
cgal/5.4.1--gcc--11.3.0                                 libmatheval/1.1.11--gcc--11.3.0                     parallel-netcdf/1.12.3--openmpi--4.1.4--gcc--11.3.0
cgal/5.4.1--openmpi--4.1.4--gcc--11.3.0                 libszip/2.1.1--gcc--11.3.0                          parallel-netcdf/1.12.3--openmpi--4.1.4--nvhpc--23.1
cineca-hpyc/2023.05                                     magma/2.6.2--gcc--11.3.0-cuda-11.8                  parmetis/4.0.3--openmpi--4.1.4--gcc--11.3.0
cudnn/8.4.0.27-11.6--gcc--11.3.0                        metis/5.1.0--gcc--11.3.0                            parmetis/4.0.3--openmpi--4.1.4--nvhpc--23.1
cutensor/1.5.0.3--gcc--11.3.0                           nccl/2.14.3-1--gcc--11.3.0-cuda-11.8                petsc/3.18.1--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
elpa/2021.11.001--openmpi--4.1.4--gcc--11.3.0-cuda-11.8 netcdf-c/4.9.0--gcc--11.3.0                        petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-complex
fftw/3.3.10--gcc--11.3.0                                netcdf-c/4.9.0--openmpi--4.1.4--gcc--11.3.0         petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--gcc--11.3.0                netcdf-c/4.9.0--openmpi--4.1.4--nvhpc--23.1         petsc/3.19.0--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--nvhpc--23.1                netcdf-fortran/4.6.0--gcc--11.3.0                   proj/8.2.1--gcc--11.3.0
gdal/3.5.3--gcc--11.3.0                                 netcdf-fortran/4.6.0--openmpi--4.1.4--gcc--11.3.0   rapids/2023.09
gsl/2.7.1--gcc--11.3.0                                  netcdf-fortran/4.6.0--openmpi--4.1.4--nvhpc--23.1   slate/2022.07.00--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
hdf5/1.12.2--gcc--11.3.0                                netlib-scalapack/2.2.0--openmpi--4.1.4--gcc--11.3.0 zlib/1.2.13--gcc--11.3.0

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/tools ------------------------------------------------------------------------
anaconda3/2022.05  emacs/28.2             gnuplot/5.4.3--gcc--11.3.0   nco/5.0.1--openmpi--4.1.4--gcc--11.3.0  singularity/3.8.7  texinfo/6.5
anaconda3/2023.03  git-lfs/3.1.2          intel-oneapi-vtune/2022.4.1  ncview/2.1.8--openmpi--4.1.4--gcc--11.3.0  snakemake/6.15.1   texinfo/6.5--gcc--11.3.0
cmake/3.24.3       git/2.38.1             jube/2.4.3                   ninja/1.11.1                            spack/0.19.1-d71   valgrind/3.19.0--openmpi--4.1.4--gcc--11.3.0
curl/7.79.0        git/2.38.1--nvhpc--23.1 maven/3.8.4                 openjdk/11.0.17_8                       superc/2.0

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/compilers -------------------------------------------------------------------------
cuda/11.8   intel-oneapi-compilers/2023.0.0     nvhpc/22.3  perl/5.36.0--gcc--8.5.0   perl/5.36.0--nvhpc--23.1   python/3.10.8--gcc--11.3.0
gcc/11.3.0  llvm/15.0.4--gcc--11.3.0-cuda-11.8  nvhpc/23.1  perl/5.36.0--gcc--11.3.0  python/3.10.8--gcc--8.5.0
```

The Module Software Environment

Available modules can be shown with the command: *module available*

```
$ module av
```

name version compiler

```
------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/profiles ------------------------------------------------------------------------
profile/archive  profile/base  profile/candidate  profile/chem-phys  profile/deeplrn  profile/lifesc  profile/meteo  profile/spoke7

------------------------------------------------------------------- /leonardo/prod/opt/modulefiles/base/libraries -------------------------------------------------------------------
adios/1.13.1--openmpi--4.1.4--gcc--11.3.0              hdf5/1.12.2--gcc--11.3.0-threadsafe                netlib-scalapack/2.2.0--openmpi--4.1.4--nvhpc--23.1
adios/1.13.1--openmpi--4.1.4--nvhpc--23.1              hdf5/1.12.2--openmpi--4.1.4--gcc--11.3.0           netlib-xblas/1.0.248--gcc--11.3.0
blitz/1.0.2--gcc--11.3.0                               hdf5/1.12.2--openmpi--4.1.4--nvhpc--23.1           openblas/0.3.21--gcc--11.3.0
boost/1.80.0--gcc--11.3.0                              intel-oneapi-mkl/2022.2.1                          openblas/0.3.21--nvhpc--23.1
boost/1.80.0--openmpi--4.1.4--gcc--11.3.0              intel-oneapi-mpi/2021.7.1                          openmpi/4.1.4--gcc--11.3.0-cuda-11.8
boost/1.80.0--openmpi--4.1.4--nvhpc--23.1              intel-oneapi-tbb/2021.7.1                          openmpi/4.1.4--nvhpc--23.1-cuda-11.8
cgal/5.4.1--gcc--11.3.0                                libmatheval/1.1.11--gcc--11.3.0                    parallel-netcdf/1.12.3--openmpi--4.1.4--gcc--11.3.0
cgal/5.4.1--openmpi--4.1.4--gcc--11.3.0                libszip/2.1.1--gcc--11.3.0                         parallel-netcdf/1.12.3--openmpi--4.1.4--nvhpc--23.1
cineca-hpyc/2023.05                                    magma/2.6.2--gcc--11.3.0-cuda-11.8                 parmetis/4.0.3--openmpi--4.1.4--gcc--11.3.0
cudnn/8.4.0.27-11.6--gcc--11.3.0                       metis/5.1.0--gcc--11.3.0                           parmetis/4.0.3--openmpi--4.1.4--nvhpc--23.1
cutensor/1.5.0.3--gcc--11.3.0                          nccl/2.14.3-1--gcc--11.3.0-cuda-11.8               petsc/3.18.1--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
elpa/2021.11.001--openmpi--4.1.4--gcc--11.3.0-cuda-11.8  netcdf-c/4.9.0--gcc--11.3.0                     petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-complex
fftw/3.3.10--gcc--11.3.0                               netcdf-c/4.9.0--openmpi--4.1.4--gcc--11.3.0        petsc/3.18.1--openmpi--4.1.4--nvhpc--23.1-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--gcc--11.3.0               netcdf-c/4.9.0--openmpi--4.1.4--nvhpc--23.1        petsc/3.19.0--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
fftw/3.3.10--openmpi--4.1.4--nvhpc--23.1               netcdf-fortran/4.6.0--gcc--11.3.0                  proj/8.2.1--gcc--11.3.0
gdal/3.5.3--gcc--11.3.0                                netcdf-fortran/4.6.0--openmpi--4.1.4--gcc--11.3.0  rapids/2023.09
gsl/2.7.1--gcc--11.3.0                                 netcdf-fortran/4.6.0--openmpi--4.1.4--nvhpc--23.1  slate/2022.07.00--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
hdf5/1.12.2--gcc--11.3.0                               netlib-scalapack/2.2.0--openmpi--4.1.4--gcc--11.3.0  zlib/1.2.13--gcc--11.3.0

------------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/tools ------------------------------------------------------------------------
anaconda3/2022.05  emacs/28.2             gnuplot/5.4.3--gcc--11.3.0   nco/5.0.1--openmpi--4.1.4--gcc--11.3.0     singularity/3.8.7  texinfo/6.5
anaconda3/2023.03  git-lfs/3.1.2          intel-oneapi-vtune/2022.4.1  ncview/2.1.8--openmpi--4.1.4--gcc--11.3.0  snakemake/6.15.1   texinfo/6.5--gcc--11.3.0
cmake/3.24.3       git/2.38.1             jube/2.4.3                   ninja/1.11.1                               spack/0.19.1-d71   valgrind/3.19.0--openmpi--4.1.4--gcc--11.3.0
curl/7.79.0        git/2.38.1--nvhpc--23.1  maven/3.8.4                openjdk/11.0.17_8                          superc/2.0

---------------------------------------------------------------------- /leonardo/prod/opt/modulefiles/base/compilers ----------------------------------------------------------------------
cuda/11.8   intel-oneapi-compilers/2023.0.0     nvhpc/22.3  perl/5.36.0--gcc--8.5.0   perl/5.36.0--nvhpc--23.1   python/3.10.8--gcc--11.3.0
gcc/11.3.0  llvm/15.0.4--gcc--11.3.0-cuda-11.8  nvhpc/23.1  perl/5.36.0--gcc--11.3.0  python/3.10.8--gcc--8.5.0
```

module: Basic Commands

Profile **base** is automatically loaded during login.

- How to **load** additional profiles (or load a module)?

```
$ module load profile/deeplrn
```

- Which modules did I already load?

```
$ module list
Currently Loaded Modulefiles:
 1) profile/base   2) profile/deeplrn
```

- How to **unload** a profile (or a module)?

```
$ module unload profile/deeplrn
```

- Or if you want to **unload all** of them

```
$ module purge
```

module: Basic Commands

Available modules can be shown with the command:

```
$ module av

------------------------------------------------------------ /cineca/prod/opt/modulefiles/deeplrn/data -------------------------------------------------------------
epic-kitchens/epic-kitchens-100  imagenet/ilsvrc2012  kinetics/kinetics400  kinetics/kinetics700

------------------------------------------------------- /cineca/prod/opt/modulefiles/deeplrn/libraries --------------------------------------------------------
cineca-ai/1.0.0              detectron2/0.2.1--python--3.8.2              netcdf/4.7.2--spectrum_mpi--10.3.1--binary  ray/1.0.0--python--3.8.2
cineca-ai/2.0.0              dlib/19.20--python--3.8.2                    nni/1.8--python--3.8.2                      ray/1.2.0--python--3.8.2
cineca-ai/2.0.1              ffmpeg/4.3--gnu--8.4.0                       open-ce/1.1.3                               scipy/1.5.1--python--3.8.2
cineca-ai/2.0.2              habitat/0.1.6--python--3.8.2                 opencv/4.3.0--gnu--8.4.0                    tensorflow/1.15.3--python--3.7.7
cineca-ai/2.1.0              hdf5/1.10.6--gnu--8.4.0                      protobuf/3.11.1--gnu--8.4.0                 tensorflow/2.3.0--cuda--10.1
cudnn/7.6.5--cuda--10.1      horovod/0.20.0--spectrum_mpi--10.3.1--binary  pytorch/1.6--cuda--10.1                     tensorflow/2.3.0--cuda--10.2
cudnn/8.0.4--cuda--10.2      jaxlib/0.1.55--python--3.8.2                 pytorch/1.6--cuda--10.2                     wmlce/1.6.2
dali/0.27.0-dev--cuda--10.2  llvmlite/0.34.0--python--3.8.2               pytorch/1.7--cuda--10.2                     wmlce/1.7.0
detectron2/0.2.1--cuda--10.2 nccl/2.7.8--cuda--10.1                       pytorch3d/0.2.5--python--3.8.2

------------------------------------------------------- /cineca/prod/opt/modulefiles/deeplrn/compilers --------------------------------------------------------
go/1.17.3

---------------------------------------------------------- /cineca/prod/opt/modulefiles/deeplrn/tools ----------------------------------------------------------
bazel/3.1.0  bazel/4.2.1  ninja/1.10.1  py-spy/0.3.3

------------------------------------------------------ /cineca/prod/opt/modulefiles/deeplrn/applications ------------------------------------------------------
deepmd/2.0  detectron2/0.1.3  jupyter/1.0.0
```

modmap command

A command automatically looks for a module in all the profiles for you

```
$ modmap -m blas
Profile: advanced
Profile: archive
Profile: astro
Profile: base
                blas
                 3.8.0–gnu--8.4.0
                openblas
                 0.3.9–gnu--8.4.0
Profile: bioinf
Profile: chem-phys
                blas
                 3.8.0–pgi--19.10--binary
                openblas
                 0.3.9–gnu--8.4.0
Profile: deeplrn
                openblas
                 0.3.9–gnu--7.2.0
Profile: lifesc
```

Loading a Module

Loading a module simply means to define or modify your **environment variables** allowing the use of the executables or the libraries.

In order to see the modifications applied by the module:

```
$ module show blas/3.8.0--gnu--8.4.0
-------------------------------------------------------------------
/cineca/prod/opt/modulefiles/base/libraries/blas/3.8.0--gnu--8.4.0:
prereq gnu/8.4.0
conflict blas
setenv BLAS_HOME /cineca/prod/opt/libraries/blas/3.8.0/gnu--8.4.0
setenv BLAS_LIB /cineca/prod/opt/libraries/blas/3.8.0/gnu--8.4.0/lib
prepend-path LIBPATH /cineca/prod/opt/libraries/blas/3.8.0/gnu--8.4.0/lib
prepend-path LD_LIBRARY_PATH /cineca/prod/opt/libraries/blas/3.8.0/gnu--8.4.0/lib
```

module help

```
$ module help cineca-ai
----------------------------------------------------------------------
Module Specific Help for /leonardo/prod/opt/modulefiles/deeplrn/libraries/cineca-ai/3.0.0:

modulefile "cineca-ai/3.0.0"
using help from /cineca/prod/opt/helps/cineca-ai/.help

------------------------------------------------------------------------------
License type: free
Web site:     https://gitlab.hpc.cineca.it/cineca-ai/cineca-ai-channel
Download url: https://gitlab.hpc.cineca.it/cineca-ai/cineca-ai-channel
------------------------------------------------------------------------------

CINECA Artificial Intelligence project

  This module has been personalised by CINECA AI experts.

- The loading of Cineca-ai module make available the most common Artificial Intelligence applications

  Tensorflow    2.10.0
  Horovod       0.26.0
  Pytorch       1.13.0
  Keras         2.10.0

  To see the complete list of all the python packages available and check their versions:

   $ module load profile/deeplrn
   $ module load cineca-ai/<version>
   $ python -m pip list

  In addition, some of the main common packages of HPC environment are automatically loaded (e.g. cuda, openMPI)

- More details about how to use cineca-ai can be found at the link:
https://wiki.u-gov.it/confluence/display/SCAIUS/Leonardo+-+Scientific+Python+user+environment+and+tools+for+AI%3A+the+CINECA+Artificial+Intelligence+project
```

Programming Environment

On Leonardo you can find different **compiler** families:

- GNU compilers: `gcc/11.3.0`
- Nvidia compilers: `nvhpc/22.3, nvhpc/23.1`
- Intel compilers: `intel-oneapi-compilers/2023.0.0`
- CUDA SDK: `cuda/11.8`

Different **MPI** implementations are available:

- `intelmpi` in `intel` compiler
- `openmpi` for `nvhpc` and `gcc` compilers
  - Support both CUDA-aware and GPUDirect technologies

During execution you need to use mpirun (not srun).

Python Virtual Environment

A Python virtual environment is a self **confinement** Python packages installation that can coexist with other installations on the same system.

venv is a module to create isolated Python environments.
venv creates a directory which contains all executables, library files and package manager to use Python packages.

Virtual environment basic usage consists in the following steps:

1. Create a virtual environment for a project
2. Activate virtual environment
3. Install packages and/or Execute commands
4. When done deactivate virtual environment

Python Virtual Environment

Load the python interpreter

```
$ module load python
```

Create your own virtual environment

```
$ python -m venv <my_venv>
```

This creates the folder my_venv in which you can find bin, include and lib folders where you will find the results of your installations.
By default venv module does not include global Python site-packages directory. To create a virtual environment including global site-packages directory add `--system-site-packages` options

```
$ python -m venv --system-site-packages <my_venv>
```

Activate your environment

```
$ source <my_venv>/bin/activate
```

Installing whatever you need (e.g numpy)

```
(my_venv) $ pip install numpy
```

Deactivating the virtualenv when you are done

```
(my_venv) $ deactivate
```

In order to use the installed packages you have to activate your environment.

Conda Environment

conda is an open source powerful package manager and environment system provided by **Anaconda** on the repo.continuum.io repository.

Anaconda is used by the **scientific** software community and it has become very popular for data science.

It is designed for data science and machine learning workflows and it is commonly used for large-scale data processing, scientific computing, and predictive analytics.

It provides a package, dependency and environment manager for **any language**:
Python, C/C++, FORTRAN, R, Ruby, Lua, Java, JavaScript.

It is available in two versions:

- **Anaconda**
  A data science platform distribution of conda. It comes with a lot of scientific python packages.
- **Miniconda**
  Lightweight distribution of conda. It only contains the necessary python packages.

Conda Environment

This is very similar to python environment but uses the module **anaconda**.

```
$ module load anaconda
```

Create your own virtual environment (you can also specify the specific version of python you would like)

```
$ conda create -n myenv python=3.10
```

This creates the folder `$HOME/.conda/envs/myenv` in which you can find all your installations.
it is better to specify a directory other than `$HOME` where to create the environment

```
$ conda create -p mypath/myenv
```

Activate your environment

```
$ conda activate myenv
```

Install whatever you need (e.g numpy)

```
(myvenv) $ conda install numpy
```

Deactivate the virtualenv when you are done working

```
(myvenv) $ conda deactivate
```

CINECA-AI Project

The AI software stack is released with **CINECA-AI Project**.

- Based on `spack` package manager.
- **New releases** for most popular software.
- This environment has been personalised by CINECA AI experts and published in a public module.
- All available packages can be loaded with **module cineca-ai**.

```
$ module load profile/deeplrn
$ module av cineca-ai
------ /leonardo/prod/opt/modulefiles/deeplrn/libraries ------
cineca-ai/2.0.0  cineca-ai/2.0.1  cineca-ai/2.1.0
cineca-ai/2.2.0  cineca-ai/3.0.0  cineca-ai/4.0.0
```

- Try to satisfy every **user request** for software installation and datasets.
  - send a mail to superc@cineca.it
- Willing to **collaborate/contribute** in AI projects supporting the community.

CINECA-AI Project

```
$ module load profile/deeplrn
$ module load cineca-ai/3.0.0

$ python -c "import torch; print(torch.__version__)"
1.13.0

$ pip list
Package                       Version              Location
----------------------------- -------------------- --------------------------------------------------
...
horovod                       0.26.1
keras                         2.10.0
mpi4py                        3.1.4
protobuf                      3.19.4
pyarrow                       10.0.1
pytorch-lightning             1.5.3
scikit-learn                  1.1.3
scipy                         1.11.1
tensorboard                   2.10.0
tensorflow                    2.10.0
tensorflow-addons             0.20.0
torch                         1.13.0
torchvision                   0.14.0
...
```

Software AI Stack @CINECA

Package	Version	Package	Version	Package	Version
Tensorflow	2.10.0	TorchText	0.14.0	Transformers	4.9.2
TensorFlow Estimators	2.10.0	TorchVision	0.14.0	Tokenizers	0.10.3
TensorFlow Probability	0.14.0	PyTorch Lightning	1.5.3	SentencePiece	0.1.91
TensorBoard	2.10.0	PyTorch Lightning Bolts	0.3.2	Spacy	3.1.2
TensorBoard Data Server	0.6.1	ONNX	1.7	Thinc	8.0.8
TensorFlow Text	2.10.0	Onnx-runtime	1.7.2	DALI	1.4.0
TensorFlow Model Optimizations	0.7.4	skl2onnx	1.9.0	OpenCV	3.4.14
TensorFlow Addons	0.20.0	tf2onnx	1.9.2	Horovod	0.26.1
TensorFlow Datasets	4.4.0	onnxmltools	1.9.1	PyArrow	10.0.1
TensorFlow Hub	0.12.0	onnxconverter-common	1.8.1	grpc	1.36.4
TensorFlow MetaData	1.10.0	XGBoost	1.4.2	protobuf	3.14
PyTorch	1.13.0	LightGBM	3.2.1	uwsgi	2.0.22

Build Based on CINECA-AI

How-To install **additional** packages within a virtual environment:

If you need a package not included in the cineca-ai modules, you can always rely on the cineca-ai environment for the dependencies and install what you need within a personal virtual environment and/or a conda environment.

```
module load profile/deeplrn
module load cineca-ai/<version>
# create your environment
python -m venv <myvenv> --system-site-packages
source <myvenv>/bin/activate
pip install PACKAGE
```

NB: the `--system-site-packages` flag gives the virtual environment access to the system site-packages directory (otherwise you cannot access the cineca-ai environment).

It is advised to create personal envs in your $WORK area, since the $HOME disk quota is limited to 50 GB.

Build Based on CINECA-AI

If you prefer working in a **conda** environment, you can activate the **CINECA-AI** env with:

```
module load profile/deeplrn cineca-ai/<version>
```

To create a personal **conda environment**:

```
module load anaconda
conda create --prefix <mycondaenv> -y
conda activate <mycondaenv>

# after activating, install the additional packages
conda install PACKAGE1
conda install -c conda-forge PACKAGE2
```

To install additional packages within a virtual environment:

```
# after activating <mycondaenv>
python -m venv <myvenv> --system-site-packages
source <myvenv>/bin/activate
pip install PACKAGE
```

Dataset Modules

Some modules with AI datasets are already available and ready to use on CINECA clusters.

These datasets are stored on a dedicated file-system optimized for fast access.

```
$ module load profile/deeplrn

$ module av
…
------------------ /cineca/prod/opt/modulefiles/deeplrn/data -------------------
epic-kitchens/epic-kitchens-100  imagenet/ilsvrc2012  kinetics/kinetics400  kinetics/kinetics700
…

$ module load imagenet

$ module show imagenet/ilsvrc2012
-------------------------------------------------------------------
/cineca/prod/opt/modulefiles/deeplrn/data/imagenet/ilsvrc2012:

conflict        imagenet
setenv          IMAGENET2012_HOME /cineca/prod/data/ai/imagenet/ilsvrc2012
module-whatis   {ImageNet ILSVRC 2012 dataset}
-------------------------------------------------------------------
```

Dataset Modules

```
$ module help imagenet/ilsvrc2012
-------------------------------------------------------------------
Module Specific Help for /cineca/prod/opt/modulefiles/deeplrn/data/imagenet/ilsvrc2012:

modulefile "imagenet/ilsvrc2012"

module name: imagenet-ilsvrc2012
creation/update date: 20210218 11:29:11

brief description: ImageNet ILSVRC 2012 dataset

----------------------------------------------------------------------------
License type: unknown
Web site:     http://www.image-net.org/
Download url: http://image-net.org/download
----------------------------------------------------------------------------

ImageNet is an image dataset organized according to the WordNet hierarchy.
Each meaningful concept in WordNet, possibly described by multiple words or
word phrases, is called a "synonym set" or "synset". There are more than
100,000 synsets in WordNet, majority of them are nouns (80,000+).
In ImageNet, we aim to provide on average 1000 images to illustrate each synset.
Images of each concept are quality-controlled and human-annotated.
In its completion, we hope ImageNet will offer tens of millions of
cleanly sorted images for most of the concepts in the WordNet hierarchy.

- The dataset is stored in $IMAGENET2012_HOME
- Contents of $IMAGENET2012_HOME: train val bbox
```

Compute Nodes: Jobs & Scheduler

What we actually want most of the time is to gain access to the compute nodes to exploit their power.

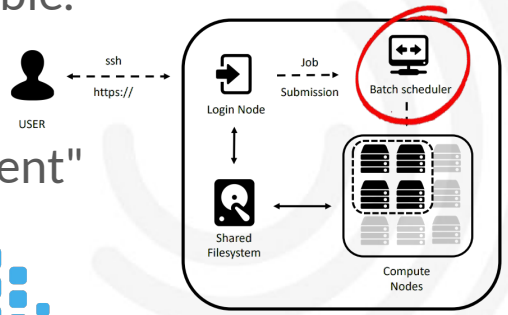
As in every HPC cluster, Leonardo allows you to run your simulations by **submitting** “jobs” to the compute nodes.

Your job is then taken in consideration by a scheduler, that adds it to a queuing line and allows its execution when the resources required are available.

The operative scheduler in Leonardo is **SLURM**.

SLURM stands for "Simple Linux Utility for Resource Management"

- Allocating access to resources
- Job starting, executing and monitoring
- Queue of pending jobs management



SLURM Resource Requirements

--nodes – number of compute nodes

--ntasks-per-node – number of tasks per node (max. 32 on Leonardo)

--cpus-per-task – number of cpus to be assigned to every task.

ntasks-per-node*cpus-per-task ≤ 32

--gres=gpu:x – number of GPUs for every node (x=1,2,3,4)

--mem – memory allocated for each node (max=512000 MB)

--exclusive – allocates nodes in exclusive mode

SLURM Directives: Walltime and Partitions

`--time=00:30:00, -t 00:30:00`

Specifies the maximum duration of the job. The **maximum time** allowed depends on the partition used.

> Pro-tip: the less walltime you ask, the faster your job will enter in execution. Think about it!

`--partition=boost_usr_prod, -p boost_usr_prod`

`--qos=boost_qos_dbg, -q boost_qos_dbg` (optional)

Specifies the "**partition**", a.k.a. the specific set of nodes among which your job can search for resources. Optionally you can specify a QoS (Quality of Service) for jobs with particular purposes, like debugging or large production

Available Partitions and QoS on Leonardo

SLURM partition	Job QOS	# cores/# GPU per job	max walltime	max running jobs per user/ max n. of cores/nodes/GPUs per user	priority	notes
boost_usr_prod	*normal*	max = 32 nodes	24:00:00		40	
	boost_qos_dbg	max = 2 nodes	00:30:00	2 nodes / 64 cores / 8 GPUs	80	
	boost_qos_bprod	min = 33 nodes max =256 nodes	24:00:00	256 nodes	60	runs on 1536 nodes min is 33 FULL nodes
	boost_qos_lprod	max = 3 nodes	4-00:00:00	3 nodes /12 GPUs	40	

see Official CINECA Docs

SLURM Jobscript Example

```
#!/bin/bash
#SBATCH -N 2                        # number of nodes to allocate
#SBATCH --ntasks-per-node=16        # number of processes per compute node
#SBATCH --cpus-per-task=2           # number of cores allocated per process
#SBATCH --gres=gpu:4                # number of gpus per node
#SBATCH -t 1:00:00                  # total run time of the job allocation
#SBATCH --mem=10GB                  # memory request per node
#SBATCH -o job.out                  # for stdout redirection
#SBATCH -e job.err                  # for stderr redirection
#SBATCH -p boost_usr_prod           # partition for resource allocation
#SBATCH -A <my_account>             # account name for allocation

module load openmpi

export OMP_PROC_BIND=true

mpirun -n 2 ./myprogram
```

SLURM Directives

`#SBATCH --job-name=myname, -J myname`

Defines the **name** of your job

`#SBATCH --output=job.out, -o job.out`

Specifies the file where the **standard output** is directed (default=slurm-<pid>)

`#SBATCH --error=job.err, -e job.err`

Specifies the file where the **standard error** is directed (default=slurm-<pid>)

`#SBATCH --mail-type=ALL (optional)`

Specifies **email** notification. An email will be sent to you when something happens to your job, according to the keywords you specified (NONE, BEGIN, END, FAIL, REQUEUE, ALL)

`#SBATCH --mail-user=user@email.com (optional)`

Specifies the e-mail address for the keyword above

SLURM Directives: Accounting

```
#SBATCH --account=<my_account>, -A <my_account>
```

Specifies the **account** to use the CPU hours from.

As an user, you have access to a limited number of CPU hours to spend. They are not assigned to users, but to projects and are shared between the users who are working on the same project (i.e. your research partners). Such projects are called accounts and are a different concept from your username.

You can check the status of your account with the command "**saldo -b**", which tells you how many CPU hours you have already consumed for each account you're assigned at (a more detailed report is provided by "saldo -r").

```
$ saldo -b
--------------------------------------------------------------------------------------------------------------------------------
account                start         end         total        localCluster    totConsumed      totConsumed      monthTotal      monthConsumed
                                                 (local h)    Consumed(local h)   (local h)             %       (local h)          (local h)
--------------------------------------------------------------------------------------------------------------------------------
cin_staff           20110323    20300323     400000004            9287758      61427173            15.4        1729106                   0
cin_QEdevel1_4      20121108    20211231         80000              73131         73131            91.4              0                   0
tra21_gpu3          20211124    20211205          4000                155           155             3.9              0                   0
```

ELLIS Summer School @LEONARDO

Account: **`tra23_ELLIS`**

Reservation:

Monday	18/09/2023	12:00 - 23:59	**`s_tra_Ellis1809`**
Tuesday	19/09/2023	12:00 - 23:59	**`s_tra_Ellis`**
Wednesday	20/09/2023	00:00 - 23:59	
Thursday	21/09/2023	00:00 - 23:59	

Only Monday: **`-A tra23_ELLIS --reservation s_tra_Ellis1809`**

All other days: **`-A tra23_ELLIS --reservation s_tra_Ellis`**

Different Views of a Compute Node

Pure **OpenMP**:

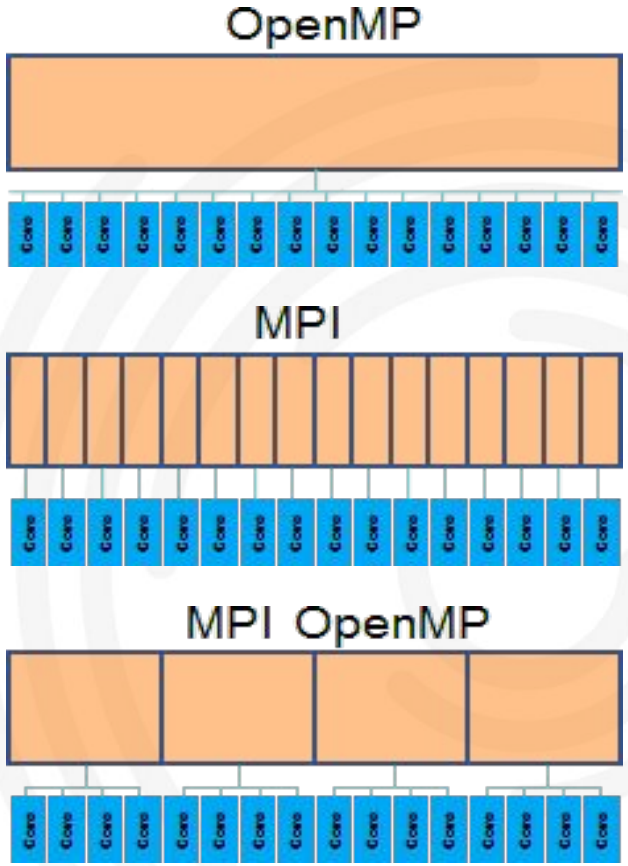
- Use OpenMP only
- Launch one process per node
- Share data using shared memory
- The master thread fork one thread per core
- Use a single core only

Pure **MPI**:

- Use MPI only
- Launch one process per core
- Share data with messages between processes
- Use one or more nodes

Hybrid **MPI/OpenMP**:

- Use both MPI and OpenMP
- Launch a given MPI process per node
- Each MPI process launches multiple OpenMP threads that can share local memory

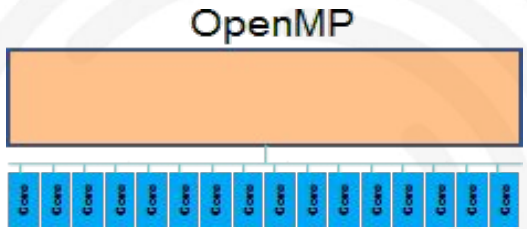


Pure OpenMP Job Example

For threaded applications (pure OpenMP, no MPI), you obtain a full node by requesting `--ntasks-per-node=1` and `--cpus-per-task=32`.

Switch the binding of the OMP threads on (default for XL compilers):

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=32     # 32 OMP threads
#SBATCH --gres=gpu:1

export OMP_PROC_BIND=true
srun --cpus-per-task=32
```



`OMP_PROC_BIND` is to ensure thread affinity (OpenMP threads spatially close to the master) and to avoid relocation of threads during the execution of a program.

There are other variables you can play with (with `OMP_PLACES` for example you can customize your binding more)

More on Thread Affinity Policy with OpenMP

OpenMP provides **`OMP_PLACES`** and **`OMP_PROC_BIND`** environment variables to specify how the OpenMP threads in a program are bound to processors:

`OMP_PROC_BIND`

It is used to specify the **binding policy** (aka thread affinity policy).
It specifies whether threads may be moved between CPUs.

- If set to **TRUE**, OpenMP threads should **not** be moved and it binds the threads to places;
- if set to **FALSE** threads may be moved and thread affinity is disabled. Use a comma separated list with the values MASTER, CLOSE and SPREAD to specify the thread affinity policy:
  - **MASTER** assign the threads in the team to the same place as the master thread.
  - **CLOSE** assign the threads in the team to the places that are close to the place of the parent thread.
  - **SPREAD** spread threads as evenly as possible among the places of the parent's place partition.

`OMP_PLACES`

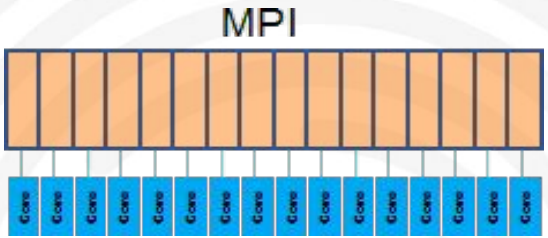
Specifies on which CPUs the threads should be placed.
The thread placement can be either specified using an abstract name or by an explicit list of the places. Allowed abstract names: threads, cores and sockets

Pure MPI Job Example

For pure multi-process application:

- Simply request 1 node with 2 processes per node and 1 GPU per process

```
#SBATCH --nodes=1              # 1 node
#SBATCH --ntasks-per-node=2    # 2 MPI processes per node
#SBATCH --gres=gpu:2           # 1 GPU per task

module load openmpi
mpirun <your exe>
```



- One MPI process per core binding processes to cores in sequential mode

```
#SBATCH --ntasks-per-node=32    # 32 MPI processes, so the full node is allocated

module load openmpi
mpirun --rank-by core -bind-to core <your exe>
```

Pure MPI Job Example

4 processes per node each process control 1 GPU.
Useful configuration for GPU applications.

- with `--ntasks-per-socket` of slurm

```
#SBATCH --ntasks-per-node=4     # 4 MPI processes per node
#SBATCH --cpus-per-task=1       # number of threads per task
#SBATCH --gres=gpu:4            # 4 gpus per task

module load openmpi
mpirun -n 4 --rank-by core -bind-to core <your exe>
```

- with `--map-by socket:PE=N` of mpirun.
  N is the number of Parallel Elements (PE), which for the object "socket" are the "physical" cores.

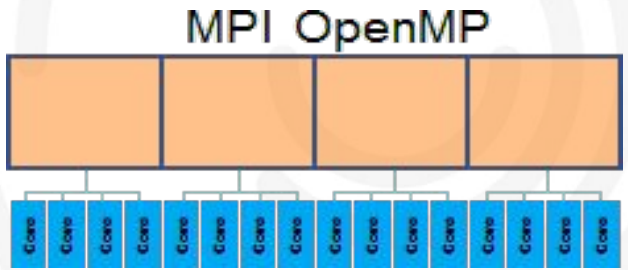
```
#SBATCH --ntasks-per-node=4     # 4 MPI processes per node
#SBATCH --cpus-per-task=1       # number of threads per task
#SBATCH --gres=gpu:4            # 4 gpus per task

module load openmpi
mpirun --map-by socket:PE=1 <your_exe>
```

Hybrid Job Example

4 processes per node, each process control 1 GPU

- Use 8 cpus per task, switch the binding of the OMP threads on, and correctly map the MPI processes with the `--map-by` option

```
#SBATCH --ntasks-per-node=4          # 4 processes per node
#SBATCH --cpus-per-task=8            # 8 cores per process
#SBATCH --gres=gpu:4                 # 4 gpus per task

export OMP_PROC_BIND=true
export OMP_NUM_THREADS=8

module load openmpi
mpirun --map-by socket:PE=8 <your_exe>
```



Submitting a Job

sbatch

```
sbatch <job_script>
```

Your job will be submitted to the SLURM scheduler and executed when there will be nodes available (according to your priority and the partition you requested).

squeue -u

```
squeue -u <username> or squeue –-me
```

Shows the list of all your scheduled jobs, along with their status (idle, running, closing, …). Also, shows you the job id required for other SLURM commands.

Interactive Batch Jobs

In case you need to “interact” with your running job (tuning of input parameters, debugging etc.) and it needs more than 10 minutes, or many processes (not suitable on the login nodes) you can submit an “Interactive” SLURM batch job.

Ask for the needed resources (cores, gpus, memory, time) with **`srun`** or **`salloc`**.

The job is queued and scheduled as any other job but, when, executed, the standard input, output, and error streams are connected to the terminal session from which srun or salloc were launched. You can then run your application from that terminal.

- **Non MPI programs** (single process or multi-threaded programs using one or more GPUs)

  ```
  $ srun ... --pty bash
  ```

  The session starts on the compute node (look at the prompt).

- **MPI programs** using one or more GPUs

  ```
  $ salloc ...
  ```

  A new session is started on the login node.

Interactive Batch Jobs

```
$ srun --nodes 1 --ntasks-per-node=1 --gres=gpu:1 -p boost_usr_prod --time=10:00 --pty /bin/bash
srun: job 6504809 queued and waiting for resources
srun: job 6504809 has been allocated resources
manpath: warning: $MANPATH set, ignoring /etc/man_db.conf
```

Look at the prompt for running job
no more `@login01 $` but the name of compute node eg. `@lrdn2529 $`

```
$ salloc --nodes 1 --ntasks-per-node=1 --gres=gpu:1 -p boost_usr_prod --time=10:00
salloc: Pending job allocation 6504817
salloc: job 6504817 queued and waiting for resources
salloc: job 6504817 has been allocated resources
salloc: Granted job allocation 6504817
salloc: Waiting for resource configuration
salloc: Nodes lrdn2529 are ready for job
manpath: warning: $MANPATH set, ignoring /etc/man_db.conf
```

The prompt has not changed, but check `$SLURM_NODELIST` environment variable for the list of nodes allocated to the job

```
$ echo $SLURM_NODELIST
lrdn2529
```

To exit from an interactive session, just type “exit” (easy to forget with salloc)

Tips&Tricks: How to Access to a Compute Node

- You are not authorised to access compute nodes through ssh connection
- Only if a node has been reserved a compute node with SLURM scheduler, can it be accessed through ssh
- This is useful for monitoring the computational resources being used (e.g. with nvidia-smi)

NB: to access a compute node, you must exchange the ssh keys between compute and login nodes.

On a login node you must execute the following commands:

```
$ ssh-keygen -t rsa -b 4096
$ cat .ssh/id_rsa.pub > .ssh/authorized_keys
```

Tips&Tricks: Jupyter Notebook On a HPC Cluster

IPython interpreter is a Python interpreter born in 2001 as a work of the student Fernando Perez.

It is based on features he liked in Mathematica and trying to create a system for everyday scientific computing (like command history, Tab auto-completion, inline editing of code, object introspection, …)

IPython notebook is an HTML-based notebook environment for Python.

It is based on the IPython shell. It provides: a web cell-based interactive environment powered with Javascript, comments and notes with HTML and markdown formats, embedded plots and other interesting features.

In 2014, Fernando Perez announced a spin-off project from IPython called Project Jupyter (**JU**lia **PYT**hon and **R**).

IPython continues to exist as a Python shell and a kernel for Jupyter, while the notebook and other language-agnostic parts of IPython have been moved under the Jupyter name.

Jupyter added support for Julia, R, Haskell and Ruby. Currently **many kernels** are available also executing compiled languages (C++, Fortran, ...).

You can start it from a terminal by running:

```
$ jupyter notebook
```

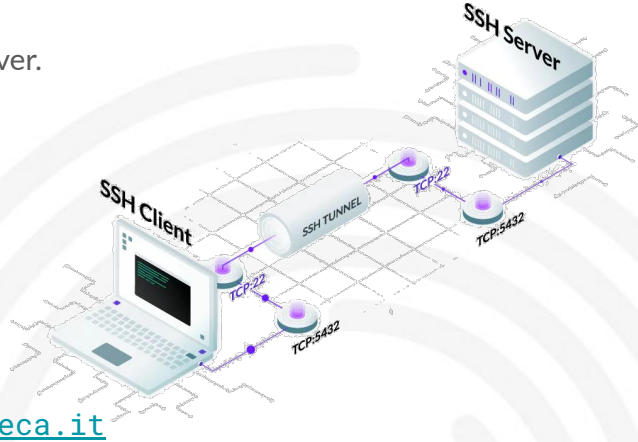
A **browser** is automatically opened with the a browsable view on your home.

Tips&Tricks: Jupyter Notebook On a HPC Cluster

SSH Port Forwarding, aka SSH tunnel, allows connection from a local computer to a remote server.
A local port can be forwarded with the `-L` option of ssh command:

```
$ ssh -N -L LOCAL:localhost:REMOTE USERNAME@REMOTE.HOST
```

NB: `-N` option is used for just forwarding ports and do not execute a remote command.



- From local machine open a SSH tunnel to remote machine, Leonardo:

  ```
  $ ssh -N -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it
  ```

- Go to remote server (Leonardo) and launch jupyter notebook without open a browser:

  ```
  $ jupyter notebook --port=9999 --no-browser
  ```

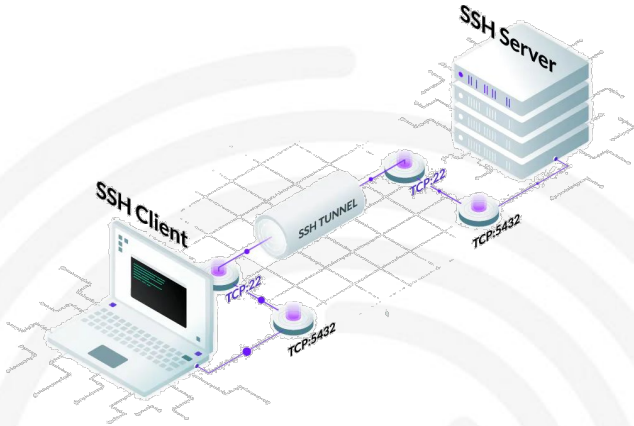
  The stdout of the command should be something like:

  ```
  http://localhost:9999/?token=75f9c6d4611a636b3249cd79fe10b218ab1f1c267d4c53d1
  ```

- Open a browser and copy&paste the URL

NB: to kill process running on a port: `$ lsof -ti:9999 | xargs kill`

Tips&Tricks: Jupyter Notebook On a HPC Cluster



1. Launch jupyter notebook on remote cluster, Leonardo:

```
$ jupyter notebook --port=9999 --no-browser
```

2. Open ssh tunnel from local machine, your laptop:

```
$ ssh -N -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it
```

3. Open notebook from a browser

```
http://localhost:9999/?token=75f9c6d4611a636b3249cd79fe10b218ab1f1c267d4c53d1
```

Tips&Tricks: Jupyter Notebook On a HPC Cluster

To open a notebook directly on a compute node a double ssh tunnel is required, aka double pike with 1 ½ somersaults and half-twist.

1. Submit an interactive job on Leonardo:

```
$ srun -N 2 --ntasks-per-node=4 --gres=gpu:4 … --pty /bin/bash

lrdn2529 $  jupyter notebook --port=9999 --no-browser
```

2. Open ssh tunnel from local machine to login node and from login node to compute node:

```
$ ssh -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it ssh -L 9999:localhost:9999 -N lrdn2529
```

3. Open notebook from a browser

```
http://localhost:9999/?token=75f9c6d4611a636b3249cd79fe10b218ab1f1c267d4c53d1
```

Tips&Tricks: Jupyter Notebook On a HPC Cluster

1. Submit an interactive job on Leonardo:

```
$ salloc -N 2 --ntasks-per-node=4 --gres=gpu:4 …
```

2. Launch jupyter notebook on remote cluster, Leonardo:

```
$ jupyter notebook --port=9999 --no-browser
```

3. Open ssh tunnel from local machine, your laptop:

```
$ ssh -N -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it
ssh -L 9999:localhost:9999 -N lrdn2529
```

4. Open notebook from a browser

```
http://localhost:9999/?token=75f9c6d4611a636b3249cd79fe10b218ab1f1c267d4c53d1
```

Tips&Tricks: How to open `tensorboard` on a cluster

1. Open ssh tunnel from local machine, your laptop:

   ```
   $ ssh -N -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it
   ```

2. Open tensorboard on remote host:

   ```
   $ tensorboard --logdir=tensorboard  --port=9999
   ```

3. Open a browser to your local machine and got to address:

   ```
   http://localhost:9999
   ```

Tips&Tricks: torch.distributed for any Launcher

```
def get_resources():

    if os.environ.get("RANK"): # launched with torchrun (python -m torch.distributed.run)
        rank = int(os.getenv("RANK"))
        local_rank = int(os.getenv("LOCAL_RANK"))
        world_size = int(os.getenv("WORLD_SIZE"))

    elif os.environ.get("OMPI_COMMAND"): # launched with mpirun
        rank = int(os.environ["OMPI_COMM_WORLD_RANK"])
        local_rank = int(os.environ["OMPI_COMM_WORLD_LOCAL_RANK"])
        world_size = int(os.environ["OMPI_COMM_WORLD_SIZE"])

    else: # launched with srun (SLURM)
        rank = int(os.environ["SLURM_PROCID"])
        local_rank = int(os.environ["SLURM_LOCALID"])
        world_size = int(os.environ["SLURM_NPROCS"])

    return rank, local_rank, world_size

rank, local_rank, world_size = get_resources()

torch.distributed.init_process_group(backend="nccl", rank=rank, world_size=world_size)

torch.cuda.set_device(local_rank)
```

Tips&Tricks: DDP on a SLURM Cluster

- List of nodes allocated to the job:

```
$ echo $SLURM_NODELIST
lrdn[2874,2877]
```

```
$ scontrol show hostnames
lrdn2874
lrdn2877
```

- Define master process:

```
$ scontrol show hostnames | head -n 1
lrdn2874
```

To run PyTorch Distributed Data Parallel script on a SLURM cluster:

```
export MASTER_ADDR=$(scontrol show hostnames "$SLURM_JOB_NODELIST" | head -n 1)
export MASTER_PORT=6000
```

Python Virtual Environment Recap

```
# load cineca-ai environment
module load profile/deeplrn
module load autoload
module load cineca-ai

# create your python environment
python -m venv --system-site-packages myenv

# load your environment
source myenv/bin/activate

# install jupyter notebook
pip install notebook
pip install matplotlib
```

ELLIS Summer School @LEONARDO

Account: **`tra23_ELLIS`**

Reservation:

Monday	18/09/2023	12:00 - 23:59	**`s_tra_Ellis1809`**
Tuesday	19/09/2023	12:00 - 23:59	**`s_tra_Ellis`**
Wednesday	20/09/2023	00:00 - 23:59	
Thursday	21/09/2023	00:00 - 23:59	

Only Monday: **`-A tra23_ELLIS --reservation s_tra_Ellis1809`**

All other days: **`-A tra23_ELLIS --reservation s_tra_Ellis`**

ELLIS Summer School @LEONARDO - SLURM

```
#!/bin/bash
#SBATCH -N 1                          # number of nodes
#SBATCH --ntasks-per-node=4           # number of processes per compute node
#SBATCH --cpus-per-task=8             # number of cores per process
#SBATCH --gres=gpu:4                  # number of gpus per node
#SBATCH -t 1:00:00                    # total run time of the job allocation
#SBATCH -o job.out                    # for stdout redirection
#SBATCH -e job.err                    # for stderr redirection
#SBATCH -p boost_usr_prod             # partition for resource allocation
#SBATCH -A tra23_ELLIS                # account name for allocation
#SBATCH --reservation s_tra_Ellis
#SBATCH --exclusive

module load profile/deeplrn
module load cineca-ai/3.0.0

export OMP_PROC_BIND=true             # bind threads to core

...
```

ELLIS Summer School @LEONARDO - SLURM

```
$ mpirun --report-bindings --map-by socket:PE=8 ./src/mind-the-bind/mind-the-bind.x

Number of nodes ............. 1
Hosts ....................... lrdn1262
Number of processes ......... 4  / SLURM_NPROCS
Number of cpus per task ..... 8  / SLURM_CPUS_PER_TASK
OMP_NUM_THREADS ............. 8

[lrdn1262:1494812] rank 0: [B/B/B/B/B/B/B/B/./././.
[lrdn1262:1494812] rank 1: [./././.
[lrdn1262:1494812] rank 2: [./././.
[lrdn1262:1494812] rank 3: [./././.

mpi_rank  0, thread_id  0 ( 0/ 0) on node lrdn1262.leonardo.local, core 0
mpi_rank  0, thread_id  1 ( 0/ 1) on node lrdn1262.leonardo.local, core 1
mpi_rank  0, thread_id  2 ( 0/ 2) on node lrdn1262.leonardo.local, core 2
mpi_rank  0, thread_id  3 ( 0/ 3) on node lrdn1262.leonardo.local, core 3
mpi_rank  0, thread_id  4 ( 0/ 4) on node lrdn1262.leonardo.local, core 4
mpi_rank  0, thread_id  5 ( 0/ 5) on node lrdn1262.leonardo.local, core 5
mpi_rank  0, thread_id  6 ( 0/ 6) on node lrdn1262.leonardo.local, core 6
mpi_rank  0, thread_id  7 ( 0/ 7) on node lrdn1262.leonardo.local, core 7
mpi_rank  1, thread_id  0 ( 1/ 0) on node lrdn1262.leonardo.local, core 8
mpi_rank  1, thread_id  1 ( 1/ 1) on node lrdn1262.leonardo.local, core 9
mpi_rank  1, thread_id  2 ( 1/ 2) on node lrdn1262.leonardo.local, core 10
mpi_rank  1, thread_id  3 ( 1/ 3) on node lrdn1262.leonardo.local, core 11
mpi_rank  1, thread_id  4 ( 1/ 4) on node lrdn1262.leonardo.local, core 12
...
```

ELLIS Summer School @LEONARDO - SLURM

```
$ srun -N 1 --ntasks-per-node=4 --cpus-per-task=8 --gres=gpu:4 -p
boost_usr_prod  -A tra23_ELLIS --reservation s_tra_Ellis
--exclusive --pty /bin/bash

srun: job 1419573 queued and waiting for resources
srun: job 1419573 has been allocated resources

lrdn1065 $ hostname
lrdn1065.leonardo.local

$ echo $SLURM_NODELIST
lrdn[1065,1071]
```

ELLIS Summer School @LEONARDO - SLURM

```
$ salloc -N 1 --ntasks-per-node=4 --cpus-per-task=8 --gres=gpu:4
-p boost_usr_prod  -A tra23_ELLIS --reservation s_tra_Ellis
--exclusive

salloc: Pending job allocation 1419565
salloc: job 1419565 queued and waiting for resources
salloc: job 1419565 has been allocated resources
salloc: Granted job allocation 1419565
salloc: Waiting for resource configuration
salloc: Nodes lrdn[2361-2362] are ready for job

login01 $ hostname
login01.leonardo.local

$ echo $SLURM_NODELIST
lrdn[2361-2362]
```

Documentation @CINECA

Read The User Guide! … and only then you can open a ticket to user support.
These are some useful links:

- Leonardo's User Guide
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.2%3A+LEONARDO+User Guide
- Work areas and filesystem
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.5%3A+Data+storage+and +FileSystems
- Accounting and budget consumption
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.4%3A+Accounting
- Batch scheduler Slurm
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.6.1%3A+How+to+submit+ the+job+-+Batch+Scheduler+SLURM